МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

С.В. Денбновецький, І.В. Мельник, Л.Д. Писаренко

КОДУВАННЯ СИГНАЛІВ В ЕЛЕКТРОННИХ СИСТЕМАХ. ЧАСТИНА 3. СПОСОБИ КОДУВАННЯ СИГНАЛІВ.

ТОМ 2. ГРУПОВІ, ІТАРАТИВНІ ТА ЗГОРТКОВІ КОДИ

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського як навчальний посібник для студентів, які навчаються за спеціальністю 171 «Електроніка» за освітніми програмами «Електронні пристрої та системи» та «Електронні прилади та пристрої»

Київ
КПІ ім. Ігоря Сікорського
2021

Рецензенти — *Забара С.С.*, доктор технічних наук, професор, завідуючий кафедрою інформаційних технологій та програмування Інституту комп'ютерних технологій Відкритого міжнародного університету розвитку людини „Україна", лауреат Державної премії СРСР, лауреат Державної премії України

Прокопенко І.Г., доктор технічних наук, професор, професор кафедри авіаційних радіоелектронних комплексів факультету аеронавігації, електроніки та телекомунікацій Національного авіаційного університету України

Відповідальний редактор — *Бідюк П.І.*, доктор технічних наук, професор

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № 5 від 14.01.2021 р.) за поданням Вченої ради факультету електроніки (протокол № 12/2020 від 21.12.2020 р.)

Електронне мережне навчальне видання

Денбновецький Станіслав Володимирович, д-р. техн. наук, проф.
Мельник Ігор Віталійович, д-р. техн. наук, проф.
Писаренко Леонід Дмитрович, д-р. техн. наук, проф.

КОДУВАННЯ СИГНАЛІВ В ЕЛЕКТРОННИХ СИСТЕМАХ. ЧАСТИНА 3. СПОСОБИ КОДУВАННЯ СИГНАЛІВ.
ТОМ 2. ГРУПОВІ, ІТАРАТИВНІ ТА ЗГОРТКОВІ КОДИ

Кодування сигналів в електронних системах. Частина 3. Способи кодування сигналів. Том 2. Групові, ітеративні та згорткові коди. [Електронний ресурс] : навч. посіб. для студ. спеціальності 171 «Електроніка», освітньої програми «Електронні пристрої та системи» та «Електронні прилади та пристрої» / С.В. Денбновецький, І.В. Мельник, Л.Д. Писаренко ; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 10,3 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2021.

У навчальному посібнику розглядаються стандартні способи кодування сигналів у сучасних електронних системах. У другому томі третьої частини посібника розглядаються групові, ітеративні та згорткові коди. Послідовно, з описанням відповідних алгоритмів кодування та декодування, розглядаються коди БЧХ, Ріда – Соломона, каскадні коди, ітераційні, згорткові коди та турбокоди. Як методи декодування послідовностей кодів Ріда – Соломона розглянуті матричні алгоритми та алгоритм Берлекемпа – Мессі, а як методи декодування згортокових кодів – алгоритми послідовного декодування та Вітербі. Відмінною рисою та несумнівною перевагою посібника є те, що в ньому наведене досконале описання алгоритмів формування різноманітних цифрових кодів, зокрема згорткових та групових. Ці алгоритми реалізовані у комп'ютерних програмах, наведених у додатках. Загалом описання алгоритмів формування групових та згорткових кодів ґрунтується на знанні математичного апарата дискретної математики, теорії груп та теорії скінченних автоматів, описаних у другій частині посібника.

Посібник призначений для студентів, які навчаються за спеціальністю «Електроніка», може бути корисним для студентів, які навчаються за спеціальностями «Телекомунікації» та «Комп'ютерна інженерія», а також для магістрів, аспірантів, викладачів та науковців відповідних спеціальностей.

Зміст

Розділ 3 Принципи побудови та способи формування групових, ітертивних та згорткових завадостійких кодів.....9
3.1 Загальні принципи формування групових кодів..... 9
3.2 Коди Файра.....14
3.3 Коди Боуза – Чоудхурі – Хоквінгема..... 25
3.3.1 Класифікація кодів Боуза – Чоудхурі – Хоквінгема та особливості їх застосування у сучасних системах зв'язку та інформаційних електронних системах.....25
3.3.2 Параметри кодів Боуза – Чоудхурі – Хоквінгема.....27
3.3.3 Алгоритм формування несистематичних кодів Боуза – Чоудхурі – Хоквінгема та відповідні приклади.....28
3.3.4 Алгоритми декодування кодів Боуза – Чоудхурі – Хоквінгема.....39
3.3.4.1 Узагальнене описання алгоритмів декодування кодів Боуза – Чоудхурі – Хоквінгема та їхня класифікація..... 39
3.3.4.2 Алгоритм Пітерсона – Горенстейна – Цирлера та процедура Ченя.....41
3.3.4.3 Алгоритм Берлекемпа – Мессі та теорема Форні…54
3.3.4.4 Алгоритм Евкліда..... 62
3.3.5 Особливості апаратної реалізації обчислювальних схем для декодерів кодів Боуза – Чоудхурі – Хоквінгема.....69
3.3.6 Цифрові електронні пристрої для формування та декодування систематичних кодів Боуза – Чоудхурі – Хоквінгема із різною коректувальною здатністю.....74
3.3.6.1 Формування та декодування систематичних кодів Боуза – Чоудхурі – Хоквінгема, які виправляють подвійні помилки.....74
3.3.6.2 Формування та декодування систематичних кодів Боуза – Чоудхурі – Хоквінгема, які виправляють потрійні помилки.....86
3.3.7 Комп'ютерна реалізація алгоритмів формування кодів Боуза – Чоудхурі – Хоквінгема та декодування

їхніх кодових послідовностей.....100
3.4 Коди Ріда – Соломона.....110
3.4.1 Загальне визначення кодів Ріда – Соломона як різновидів кодів Боуза – Чоудхурі – Хоквінгема.....110
3.4.2 Параметри кодів Ріда – Соломона.....112
3.4.3 Відображення кодів Ріда – Соломона на двійкові коди.....116
3.4.4 Основні алгебраїчні та поліноміальні операції у полях Галуа $GF(2^m)$.....118
3.4.4.1 Алгебраїчні операції над елементами полів Галуа $GF(2^m)$ та їхня програмна реалізація у системі MatLab.....118
3.4.4.2 Поліноміальні операції у полях Галуа $GF(2^m)$ та їхня програмна реалізація у системі MatLab.....140
3.4.5 Алгоритм формування кодів Ріда – Соломона та відповідні приклади.....153
3.4.6 Декодування кодів Ріда – Соломона з використанням алгоритму Берлекемпа — Мессі та теореми Форні та відповідні приклади.....162
3.4.7 Пошук коефіцієнтів полінома локаторів помилок через матричні перетворення та через обчислення визначників.....191
3.4.8 Порівняльний аналіз алгоритму Берлекемпа – Мессі з матричними алгоритмами та обрання способу написання програми, призначеної для декодування послідовностей кодів Ріда – Соломона…….195
3.4.9 Описання особливостей реалізації програми, призначеної для формування та декодування послідовностей кодів Ріда – Соломона……...199
3.4.10 Двійкові коди Ріда – Соломона та особливості їхнього формування та декодування.....209
3.4.11 Узагальнене описання розглянутих способів формування систематичних кодів Ріда – Соломона та декодування їхніх послідовностей.....217
3.4.12 Апаратні цифрові електронні пристрої для формування

кодів Ріда – Соломона та декодування їхніх послідовностей.....221
3.4.13 Оцінка залежності коректувальної здатності кодів Ріда – Соломона від довжини кодової послідовності та від порядку поля Галуа.....227
3.5 Каскадні коди…….....237
3.6 Спектральні методи формування групових кодів.....246
3.6.1 Визначення дискретного перетворення Фур'є у полі Галуа та його властивості……...246
3.6.2 Обмеження спряженості для елементів дискретного спектру……..248
3.6.3 Сліди та їхні властивості ……..250
3.6.4 Спектральне подання циклічних кодів та кодів Ріда – Соломона……..252
3.6.5 Розширені коди Ріда – Соломона…….256
3.7 Ітеративні коди…….....259
3.8 Згорткові коди……..……...266
3.8.1 Загальний принцип формування згорткових кодів.....266
3.8.2 Параметри згорткових кодів та структурна схема кодеру……...269
3.8.3 Приклади формування згорткового коду…….271
3.8.4 Головні способи подання згорткових кодів……277
3.8.4.1 Поліноміальне подання…….……...277
3.8.4.2 Подання кодера згорткового коду у вигляді діаграми станів скінченного автомату…….....279
3.8.4.3 Деревоподібні діаграми…….....282
3.8.4.4 Ґраткові діаграми…….....284
3.8.5 Метрики шляхів згорткового коду та пошук помилок за метриками.....287
3.8.6 Алгоритми декодування згорткових кодів….....296
3.8.6.1 Послідовне декодування….....296
3.8.6.2 Декодування із зворотним зв'язком.....304
3.8.6.3 Декодування за принципом максимальної правдоподібності та алгоритм Вітербі…….....308

3.8.6.4 Порівняльний аналіз алгоритмів Вітербі та послідовного декодування…….....317
3.8.7 Матричне подання згорткових кодів……..320
3.8.8 Прості структурні схеми кодерів та декодерів згорткових кодів……...326
3.8.9 Ускладнені конструкції згорткових кодів для виправлення пакетних помилок та методи їх формування…..……....330
3.8.10 Алгоритм Фано та його апаратна реалізація на логічних схемах та регістрах зсуву……337
3.8.11 Апаратна реалізація кодерів та декодерів згорткових кодів, призначених для вирішення практичних завдань електроніки…….349
3.8.11.1 Прості цифрові електронні схеми кодерів та декодерів згорткового коду на регістрах зсуву…….349
3.8.11.2 Апаратна реалізація кодерів та декодерів згорткових кодів, призначених для запису цифрової інформації на магнітні та оптичні носії...351
3.8.12 Програмна реалізація кодерів та декодерів згорткових кодів з різними параметрами……...356
3.8.12.1 Матриці станів згорткового коду та їх алгоритмічне описання через числові тривимірні масиви……...356
3.8.12.2 Приклади формування матриці станів згорткового коду…….....361
3.8.12.3 Узагальнений алгоритм формування послідовностей згорткового коду…….…….370
3.8.12.4 Алгоритм пошуку шляху за ґратковою діаграмою згорткового коду через аналіз матриці станів……373
3.8.12.5 Алгоритм виправлення помилок у послідовностях згорткових кодів через

пошук шляху із найменшою метрикою……378
3.8.12.6 Особливості написання комп'ютерної програми, призначенної для формування та розшифрування послідовностей згорткових кодів за описаними алгоритмами…….383
3.8.13 Узагальнені оцінки коректувальних параметрів згорткових кодів……391
3.8.13.1 Поняття про мінімальний просвіт згорткового коду…….391
3.8.13.2 Обчислення мінімального просвіту з використанням теорії скінченних автоматів……395
3.8.13.3 Лічильники гілок та узагальнена форма передавальної функції декодеру згорткового коду……...400
3.8.13.4 Приклади отримання аналітичних виразів для передавальних функцій згорткових кодів з іншими параметрами……..……....403
3.8.13.5 Порівняльний аналіз методів ручного та комп'ютерного розрахунку мінімального просвіту згорткового коду……..455
3.8.13.6 Залежність коректувальної здатності згорткового коду від передавальної функції та параметру мінімального просвіту..……...461
3.9 Турбокоди……..……..469
3.9.1 Основи теорії побудови турбокодів……..469
3.9.2 Алгоритм ітеративного декодування турбокодів…..471
3.9.3 Приклад формування та розшифрування ітеративного турбокоду……475
3.9.4 Формування турбокодів на основі згорткових кодів……478
3.10 Порівняльний аналіз завадостійких кодів та перспективи їхнього подальшого розвитку…….....481

Контрольні питання та завдання до розділу 3.....483
Перелік посилань…….522
Додаток Л. Програма для формування коду Файра та декодування його кодових послідовностей…….....529
Додаток М. Програма для формування систематичних кодів Боуза – Чоудхурі – Хоквінгема (15, 7) та (15, 5) та декодування їхніх кодових послідовностей…….537
Додаток Н. Програмні модулі для реалізації алгебраїчних та поліноміальних операцій у полях Галуа GF(2^m)……...548
Додаток О. Програма для формування систематичних кодів Ріда – Соломона та декодування їхніх кодових послідовностей……576
Додаток П. Програма для формування згорткових кодів з різними параметрами та декодування їхніх кодових послідовностей……..600
Додаток Р. Розрахунок поліноміальних коефіцієнтів для передавальної функції скінченного автомату, який відповідає моделі згорткового коду з параметрами K = 5, n = 1/3, з використанням функцій символьного процесора системи науково-технічних рохрахунків MatLab……..……..……..……619

Розділ 3 Принципи побудови та способи формування групових, ітеративних та згорткових завадостійких кодів

У цьому розділі розглядаються загальні принципи формування групових кодів, зокрема кодів Файра, Боуза – Чоудхурі – Хоквінгема, Ріда – Соломона та каскадних кодів, а також ітеративних та згорткових кодів, зокрема кодів Вітербі. Розглядаються також основні принципи формування турбокодів. Наведені прості приклади формування блокових та згорткових кодів та комп'ютерні програми, призначені для їхнього формування та декодування.

3.1 Загальні принципи формування групових кодів

Перед вивченням цього підрозділу необхідно повторити розділи 2 та 3 другої частини посібника

Розглянуті у підрозділах 2.4 та 2.5 лінійні та циклічні коди, згідно із загальними принципами завадостійкого кодування, описаними у підрозділі 2.2, дозволяють виявляти та виправляти помилки заданої кратності. У цьому випадку кількість помилок, які виявляються та виправляються, визначається через мінімальну кодову відстань згідно із співвідношеннями (2.35) – (2.37), наведеними у другій частині посібника. Недоліком таких способів кодування є залежність кількості помилок, які виявляються та виправляються, від методів формування коду, що не дозволяє виправляти пачки помилок у разі, якщо їх кількість у пачці та кількість пачок заздалегідь є невідомою. У цьому і полягає більш складне, узагальнене завдання завадостійкого кодування, для розв'язування якого використовуються групові та згорткові коди. Описанню способів формування таких кодів буде присвячений цей розділ посібника.

Загалом принципи формування групових кодів базуються на теорії груп та полів Галуа, яка була розглянута у другому розділі другої частини посібника [48], а принципи формування ітеративних згорткових кодів базуються на теорії імовірності та теорії дерев, які були розглянуті у шостому та сьомому розділах другої частини посібника [49, 50].

Принцип формування блокових завадостійких кодів загалом базується на теорії поліномів над групами та циклотомічних класів, яка була розглянута у третьому розділі другої частини посібника. Загальна теорія формування блокових кодів у найбільш повному вигляді вперше була розглянута у класичних монографіях відомих американських математиків Е. Берлекемпа [52] та У. Пітерсона [62]. Також методи формування групових кодів просто та дохідливо описані у підручниках [33, 55 – 57, 63, 67, 78, 79].

Елвін Берлекемп
Народився в 1940 р.

Загалом, згідно із теоретичними міркуваннями, наведеними у монографії Е. Берлекемпа [52], принцип формування блокових групових кодів можна описати наступним чином. Кожна позиція лінійного коду, наприклад коду Хеммінга, нумерується через контрольні суми, аналогічні (2.58) – (2.62), наведеними у підрозділі 2.4, і, таким чином, синдром помилки може бути обчисленим через перевірочну матрицю **H**, задану співвідношенням (2.68). Для обчислення синдромів подвійних помилок у лінійних кодах використовуються більш складні обчислювальні алгоритми, які були описані у підрозділі 2.4.2. Перехід від лінійних до блокових кодів, згідно із теорією Е. Берлекемпа, полягає у тому, що замість простих алгебраїчних операцій над бітами інформаційного слова використовують алгебраїчні операції над групами, які були розглянуті у другому та третьому розділах другої частини посібника. Задача полягає у тому, щоб модифікувати перевірочну матрицю завадостійкого коду таким чином, щоб результат її множення та спотворену

кодову комбінацію відображував не вектор помилки, а набір векторів, який описує синдроми помилок у всіх розрядах. Такий набір векторів формується через векторні операції, основу яких складає теорія груп та теорія поліномів над групами, описані у другому та третьому підрозділах другої частини посібника. За таких умов алгебраїчні операції над елементи груп, які використовуються для формування групового коду, мають задовольняти умовам зімкненості, асоціативності та комутативності [52, 62, 63, 67, 78, 79].

Розглянемо спосіб формування синдромів помилок з використанням класів лишків поліномів над групами. Тут важливою є теорія циклотомічних класів, яка була розглянута у підрозділі 3.6.10 другої частини посібника. Важливим також є те, що степені коренів поліномів циклічно повторюються, і саме це дозволяє формувати синдроми помилок.

Розглянемо простий приклад формування синдрому помилок у груповому коді, наведений у монографії Е. Берлекемпа. Припустимо, що корені поліному, який відповідає спотвореній комбінації, β_1 та β_2, відповідають номерам спотворених символів. Якщо числа β_1 та β_2 записані у двійковій формі, їх можна розглянути як класи лишків двійкового поліному $S(x)$, який описує спотворену кодову послідовність, відносно твірного поліному $g(x)$. У цьому разі встановлюється гомоморфне відношення:

$$\beta_i \leftrightarrow \beta^{(i)}(x), \tag{3.1}$$

де $\beta^{(i)}(x)$ – двійкові поліноми, степінь яких є меншою за степінь твірного полінома $g(x)$. Згідно із методом формування синдромів помилок, поліноми $\beta^{(i)}(x)$ є остачами від ділення поліному $S(x)$ на твірний поліном $g(x)$, тобто:

$$\beta^{(i)}(x) = \mathrm{mod}\,(S(x), g(x)). \tag{3.2}$$

Запишемо перевірочну матрицю завадостійкого коду наступним чином. Перші п’ять рядків матриці відповідають матриці коду Хеммінга (5, 31), а наступні п’ять – виправленню другої помилки. Відповідно, рядки перевірочної матриці з шостого по десятий, формуються через степеневі функції $f_k(x)$. Тоді перевірочна матриця групового коду **H** буде мати наступний вигляд [52]:

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \dots & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & \dots & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & \dots & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & \dots & 0 & 1 \\ f_1(1) & f_1(2) & f_1(3) & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ f_2(1) & f_2(2) & f_2(3) & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ f_3(1) & f_3(2) & f_3(3) & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ f_4(1) & f_4(2) & f_4(3) & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ f_5(1) & f_5(2) & f_5(3) & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix}. \quad (3.3)$$

Тепер головною задачею формування групового коду є пошук поліномів $f_1(x) - f_5(x)$. Перші п'ять рівнянь матриці (3.3) є лінійними рівняннями відносно змінних β_1 та β_2, тобто:

$$\beta_1 + \beta_2 = \xi_1. \quad (3.4)$$

Друге рівняння, яке відповідає контрольним перевіркам та формує синдром помилки, зважаючи на те, що функції $f_1(x) - f_5(x)$ є степеневими, запишемо у вигляді:

$$f(\beta_1) + f(\beta_2) = \xi_2. \quad (3.5)$$

Згідно із теорією груп функції $f(\beta)$ не можуть бути лінійними комбінаціями змінних β_1 та β_2, оскільки за такої умови рядки матриці (3.3) будуть взаємозалежними.

Розглянемо тепер випадок $f(\beta) = \beta^2$. За такої умови система рівнянь (3.4), (3.5) переписується у вигляді:

$$\begin{cases} \beta_1 + \beta_2 = \xi_1; \\ \beta_1^2 + \beta_2^2 = \xi_2. \end{cases} \quad (3.6)$$

Проте нескладні алгебраїчні розрахунки показують, що система рівнянь (3.6) також є взаємозалежною. Дійсно [52]:

$$\xi_1^2 = (\beta_1 + \beta_2)^2 = \beta_1^2 + 2 \cdot \beta_1 \cdot \beta_2 + \beta_2^2 = \beta_1^2 + \beta_2^2 = \xi_2.$$

У зв'язку з цим розглянемо степеневу функцію більш високого порядку. Будемо вважати, що $f(\beta) = \beta^3$. Тоді система рівнянь для формування перевірочної матриці буде мати вигляд [52]:

$$\begin{cases} \beta_1 + \beta_2 = \xi_1; \\ \beta_1^3 + \beta_2^3 = \xi_2. \end{cases} \tag{3.7}$$

Систему рівнянь (3.7) можна переписати у вигляді наступного степеневого алгебраїчного рівняння [52]:

$$\xi_2 = \beta_1^3 + \beta_2^3 = (\beta_1 + \beta_2) \cdot \left(\beta_1^2 - \beta_1 \cdot \beta_2 + \beta_2^2\right) = \xi_1 \cdot \left(\beta_1^2 + \beta_1 \cdot \beta_2 + \beta_2^2\right) =$$

$$= \xi_1 \cdot \left(\beta_1 \cdot \beta_2 - \xi_1^2\right). \tag{3.8}$$

Тобто, згідно із отриманою залежністю (3.8), систему рівнянь (3.7) можна переписати у вигляді [52]:

$$\begin{cases} \beta_1 + \beta_2 = \xi_1; \\ \beta_1 \cdot \beta_2 = \xi_1^2 + \dfrac{\xi_2}{\xi_1}. \end{cases} \tag{3.9}$$

Із співвідношення (3.9) слідує, що змінні β_1 та β_2 задовольняють алгебраїчному рівнянню [52]:

$$\beta(\xi_1 + \beta) = \xi_1^2 + \frac{\xi_2}{\xi_1}, \Rightarrow \beta^2 + \xi_1 \cdot \beta + \left(\xi_1^2 + \frac{\xi_2}{\xi_1}\right) = 0, \Rightarrow$$

$$\Rightarrow 1 + \xi_1 \cdot \beta^{-1} + \left(\xi_1^2 + \frac{\xi_2}{\xi_1}\right) \cdot \beta^{-2} = 0. \tag{3.10}$$

Розглянемо окремо випадки одиночної помилки та відсутності помилки. Для одиночної помилки значення β_2 дорівнює нулю, тому систему рівнянь (3.7) можна переписати наступним чином [52]:

$$\beta + \xi_1 = 0, \Rightarrow 1 + \xi_1 \beta^{-1} = 0. \tag{3.11}$$

Зрозуміло також, що за умови відсутності помилки її синдром дорівнює нулю, тобто:

$$\xi_1 = \xi_2 = 0. \tag{3.12}$$

Проте головна особливість способів формування групових кодів, зокрема кодів Боуза – Чоудхурі – Хоквінгема та Ріда – Соломона, які розглядатимуться у підрозділах 3.3 та 3.4, полягає у тому, що для їхньої побудови зручніше використовувати не безпосередньо поліном (3.10), а

зворотний до нього поліном, корені якого є оберненими групами до синдромів помилки [52, 62]. Такий поліном називається поліномом синдрому помилки та записується наступним чином [52, 62]:

$$\sigma(z) = \prod_{\beta} (1 - \beta z). \tag{3.13}$$

Часто для позначення сум синдромів помилок за різними степенями використовують символ S_k. Тоді для синдромів помилок можна записати наступні вирази [52, 62]:

$$S_1 = \sum \beta = \xi_1, \qquad S_2 = \sum \beta^3 = \xi_2. \tag{3.14}$$

З урахуванням (3.13), вирази (3.14) для синдромів помилок можна переписати наступним чином [52, 62]:

$$\sigma(z) = \begin{cases} 1, & \text{за умови відсутності помилки;} \\ 1 + S_1 z, & \text{за умови однієї помилки;} \\ 1 + S_1 z + \left(S_1^2 + \dfrac{S_3}{S_1} \right) \cdot z^2, & \text{за умови двох помилок.} \end{cases} \tag{3.15}$$

Тобто, з використанням співвідношень (3.15) можна шукати одиничні та підвійні помилки у двійкових та багатопозиційних кодових послідовностях. Більш досконало способи формування групових завадостійких кодів розглядатимуться на конкретних прикладах у підрозділах 3.2, 3.3 та 3.4. Найбільш поширеними серед групових кодів, які широко використовуються у сучасних системах зв'язку та електронних системах, є коди Боуза – Чоудхурі – Хоквінгема, окремим випадком яких є коди Ріда – Соломона.

3.2 Коди Файра

Перед вивченням цього підрозділу необхідно повторити підрозділ 1.1, розділ 2 та підрозділ 3.6 другої частини посібника

Найпростішими із групових кодів, призначених для виявлення та виправлення помилок заданої кратності, незалежно від їх розташування у кодовій комбінації, є коди Файра [5]. Спосіб побудови кодів Файра базується

на алгоритмах формування циклічних кодів, розглянутих у підрозділі 2.5, відмінність полягає лише у методі визначення твірного поліному. Також існує певна особливість побудови декодувальних схем для кодів Файра, яка полягає в тому, що детектор помилки другої схеми ділення автоматично налаштовується на необхідний виділений синдром, для чого і використовується, другий, відповідним чином підібраний поліном [5].

Тобто, твірний поліном $g(x)$ для циклічних кодів, призначених для виправлення помилок заданої кратності, записується у вигляді [5]:

$$g(x) = g_1(x) \cdot g_2(x). \tag{3.16}$$

Головним правилом під час використання співвідношення (3.1) є те, що поліноми $g_1(x)$ та $g_2(x)$ мають бути взаємно простими.

Зокрема, для кодів Файра [5]:

$$g(x) = p(x) \cdot (x^{2b-1} + 1), \tag{3.17}$$

де $p(x)$ – незвідний поліном степені $m \geq b$, який належить степені $e = 2^m - 1$, b – довжина пакета помилок, який можна виправляти.

Для розуміння принципу роботи кодів Файра розглянемо узагальнену систему зв'язку, наведену у розділі 2.3 першої частини посібника, та запишемо цифровий сигнал $h(x)$, який надходить на вхід приймача, у вигляді суми за модулем два неспотвореної кодової комбінації $f(x)$ та вектора $x^j \cdot B(x)$, якому у загальному випадку відповідає пачка помилок $B(x)$ [5]:

$$h(x) = f(x) \otimes x^j \cdot B(x), \tag{3.18}$$

де множник x^j описує положення пачки помилок у кодовій комбінації $h(x)$.

Із співвідношень (3.16) та (3.18) видно, що вектор $f(x)$ без остачі ділиться на поліноми $g_1(x)$ та $g_2(x)$, тому процес декодування кодів Файра можна обмежити аналізом вектора $x^j \cdot B(x)$. Необхідно також відмітити, що за умови обраної кількості розрядів у пакеті помилок та степені твірного поліному $m = b$, сукупність різноманітних пакетів помилок, які виправляються, гомоморфно відображається на сукупність остач від ділення поліному $f(x)$ на поліном $p(x)$. Теорія гомоморфних відображень як елемент теорії груп та теорії множин була описана у підрозділі 2.4 другої частини посібника [48].

Зазвичай код Файра будується наступним чином. Як остачу від ділення $f(x)/p(x)$ на такті n бажано мати синдром помилки, який відповідає вектору помилки [5, 52, 53, 62, 63]. Тоді, на наступних тактах, після введення цієї остачі, разом із спотвореною кодовою комбінацією $h(x)$, до буферного регістру пам'яті, спотворені символи легко виправляються через сумування за модулем два вектора помилки та спотвореної кодової комбінації.

Відповідна принципова електрична схема, призначена для формування коду Файра, наведена на рис. 3.1, а схема декодера кодів Файра показана на рис. 3.2 [5, 33].

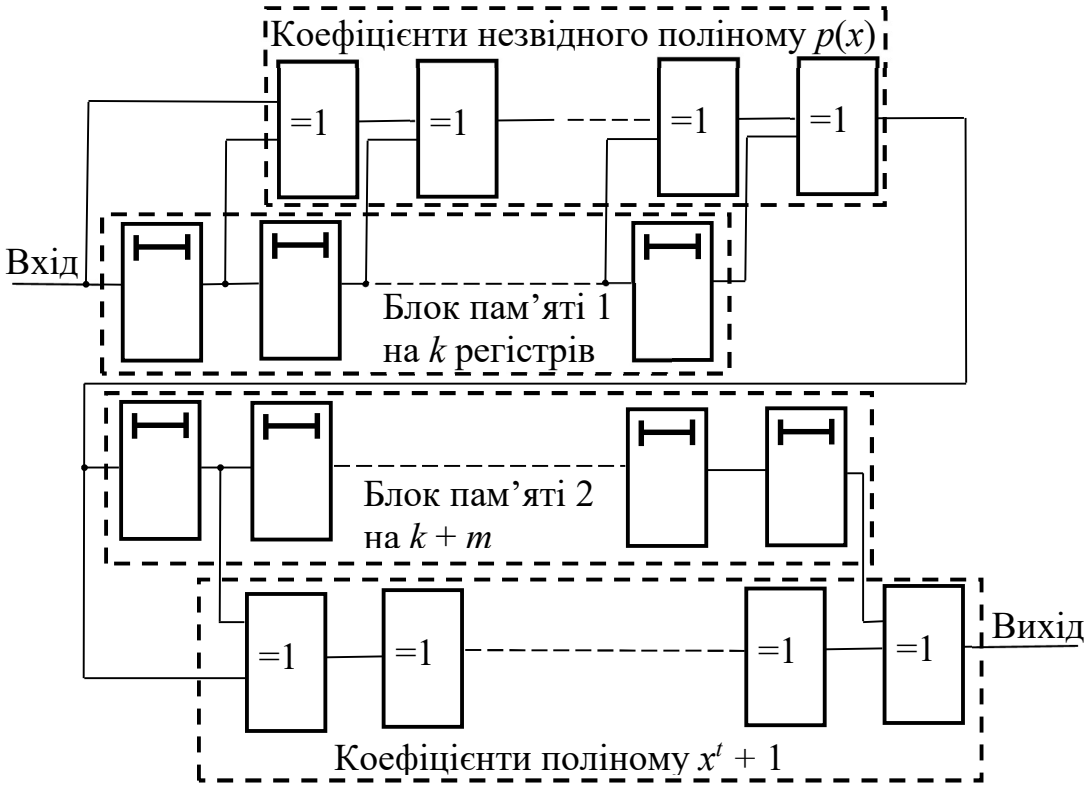


Рис. 3.1 Схема формування кодів Файра

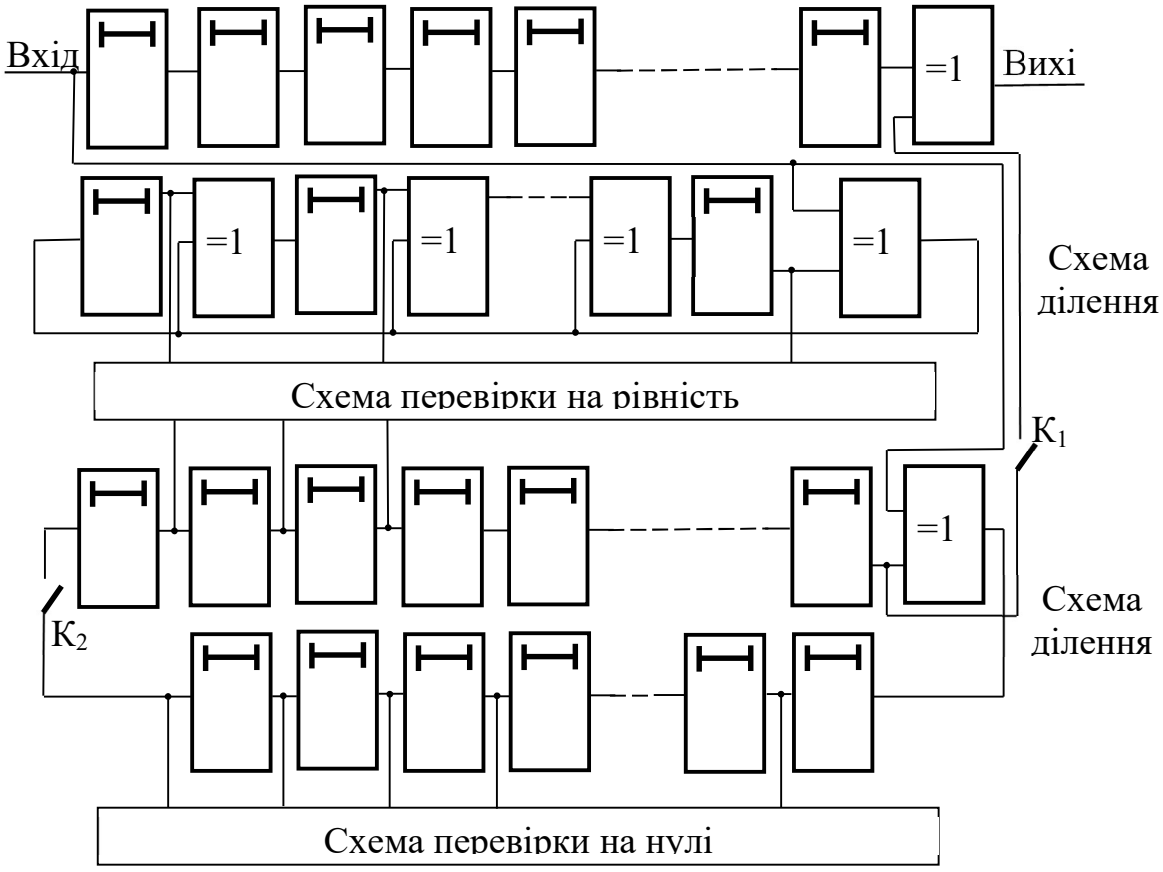


Рис. 3.2 Схема декодера кодів Файра

Схема кодера кодів Файра базується на основі узагальненої схеми множення вхідної послідовності на твірний поліном, яка була наведена на рис. 2.41, а. Слід відзначити, що ця схема відповідає зворотному порядку надходження бітів кодової послідовності. У разі прямого порядку надходження бітів для виконання операції множення слід використовувати іншу схему, наведену на рис. 2.41, б.

Відмінною рисою схеми для формування кодів Файра, наведеної на рис. 3.1, є те, що в ній використовується дві схеми множення із відповідною кількістю регістрів пам'яті [5]. Згідно із співвідношенням (3.17), початкова послідовність бітів, яка кодується, спочатку множиться на твірний поліном $p(x)$, а після цього – додатково на поліном $x^t + 1$, де $t = 2b - 1$. Для першої схеми множення необхідна кількість регістрів пам'яті становить k, де k – кількість інформаційних розрядів, а для другої схеми множення кількість елементів пам'яті є більшою і складає $k + m$.

Проаналізуємо тепер особливості схеми декодера, наведеної на рис. 3.2.

Зрозуміло, що, згідно із співвідношенням (3.18), остача від ділення на такті n буде отримана лише у тому випадку, коли ділення проводиться з першого такту і поліном $h(x)$ множиться на x^m. Такий алгоритм, пов'язаний із послідовним множенням та діленням та із порівнянням остач, схожий на алгоритм пошуку помилки у систематичних циклічних кодах, який був розглянутий у підрозділі 2.5.5 та блок-схема якого наведена на рис. 2.37.

Згідно із схемою, наведеною на рис. 3.2, під час надходження кодової послідовності $h(x)$ на першу схему ділення в ній починається закономірне чередування остач. Синдром помилки $B(x)$ є однією з таких остач, яка вперше з'являється на такті з номером $2^m - 1$. Зрозуміло, що в процесі ділення поліному $h(x)$ на поліном $x^t + 1$, згідно із співвідношенням (3.17), виникає t остач, оскільки поліном $x^t + 1$ належить своєму показнику степені t. Вектор остачі $B(x) \cdot x^{b-1}$ вперше виникає на такті з номером t, і ця остача циклічно повторюється з періодом t. Це цілком відповідає теорії циклотомічних класів лишків, яка розглядалася у підрозділі 3.6.10 другої частини посібника.

Якщо є потреба записати цю остачу від ділення до детектора помилки на такті n, необхідно, щоб число n було кратним t. Необхідно також, щоб детектор помилки не спрацював раніше. Для забезпечення виконання цієї умови, згідно із теорією чисел, розглянутою у підрозділі 1.1 другої частини посібника, числа t та e мають бути взаємно простими, а число n – їхнім найменшим спільним кратним (НСК). Тоді рівність остач $B(x)$ у регістрах двох схем ділення, а також рівність нулю інших розрядів регістру схеми ділення 2, є достатніми умовами для виявлення та виправлення пакетної помилки.

Іншою особливістю роботи декодувальної схеми, наведеної на рис. 3.2, є наявність двох ключів $К_1$ та $К_2$. Після фіксації умови, яка відповідає виправленню помилки, ключ $К_1$ замикається, а ключ $К_2$ розмикається. На схему корекції, яка побудована на основі суматора за модулем два, одночасно надходять сигнали синдрому помилки $B(x)$ з буферного регістру та зі схеми ділення 2, за рахунок чого і усується помилка у векторі спотвореної кодової послідовності $h(x)$.

У загальному випадку, коли синдром помилки $B(x)$ починається не зі старшого розряду, а з розряду із номером j, необхідно домножати кодову комбінацію $h(x)$ на поліном x із визначенням остач за модулем $p(x)$ у першій схемі ділення та за модулем $x^t + 1$ у другій схемі ділення. Спосіб визначення остач за заданим модулем для визначених поліномів розглядався у підрозділі 3.6.4. Тоді, для забезпечення рівності остач, необхідно зробити $n - j$ додаткових тактів, у цьому разі значення j може змінюватись в діапазоні від 1 до n. За $n - j$ наступних тактів вміст буферного регістру зміщується і помилкові символу знову розташовуються безпосередньо перед схемою корекції.

Якщо за час проходження кодової комбінації $h(x)$ співпадання остач від її ділення на поліноми $p(x)$ та $x^t + 1$ не виникає, вважається, що помилки, які виникли, на виправляються. За умови, якщо вже на першому такті обидві остачі від ділення дорівнюють нулю, вважається, що кодова комбінація $h(x)$ передана безпомилково.

Згідно із наведеними відомостями алгоритм формування коду Файра можна записати наступним чином [5].

1. Згідно із таблицею, наведеною у додатку І, обираємо твірний поліном

степені m, де m – просте число.

2. Визначаємо кількість помилок b, які необхідно виправляти у пакеті. Необхідні умови: $m \geq b$; $t = 2b - 1$ – просте число; m та t – взаємно прості числа.

3. Обчислення кодової комбінації $h(x)$ згідно із співвідношенням (3.17). Кількість розрядів k в інформаційному слові $f(x)$ має бути меншою за максимальну величину $k_{\max} = (2^m - 1) \cdot t - (m + t)$.

Узагальнений алгоритм декодування послідовностей коду Файра можна у загальному вигляді записати наступним чином.

1. Номер ітерації $N = 0$. Кодова послідовність $s_0(x) = h(x)$.

2. Кодова послідовність $s_N(x)$ ділиться на твірний поліном $p(x)$. Остача від ділення $r_N(x)$.

3. Кодова послідовність $s_N(x)$ ділиться на поліном $w(x)=x^t + 1$. Остача від ділення $q_N(x)$.

4. Із остачі $q_N(x)$ видаляється послідовність останніх нулів.

5. Якщо $r_0(x) = q_0(x) = 0$ – помилки немає. Перехід до пункту 9 алгоритму.

6. Якщо $r_N(x) = q_N(x)$ – знайдений синдром помилки. Перехід до пункту 8 алгоритму.

7. Якщо $r_N(x) \neq q_N(x)$ – тоді: $N = N + 1$; $s_{N+1}(x) = s_N(x) \cdot x$, та повернення до пункту 2 алгоритму. Тобто, до кодової послідовності $s_N(x)$ приписується 0 до молодшого розряду та здійснюється перехід до наступної ітерації $N + 1$.

8. Починаючи з розряду N, який рахується зліва, до кодової комбінації $h(x)$ додається за модулем два остача $r_N(x)$. Після цього кодова комбінація $H(x) = h(x) \otimes R_N(x)$ вважається виправленою.

9. Кінець проведення обчислень.

Зрозуміло, що описаний алгоритм декодування кодів Файра є дуже схожим на алгоритм декодування систематичних циклічних кодів, блок-схема якого була наведена у підрозділі 2.5 на рис. 2.37. Блок-схема алгоритму декодування кодів Файра наведена на рис. 3.3.

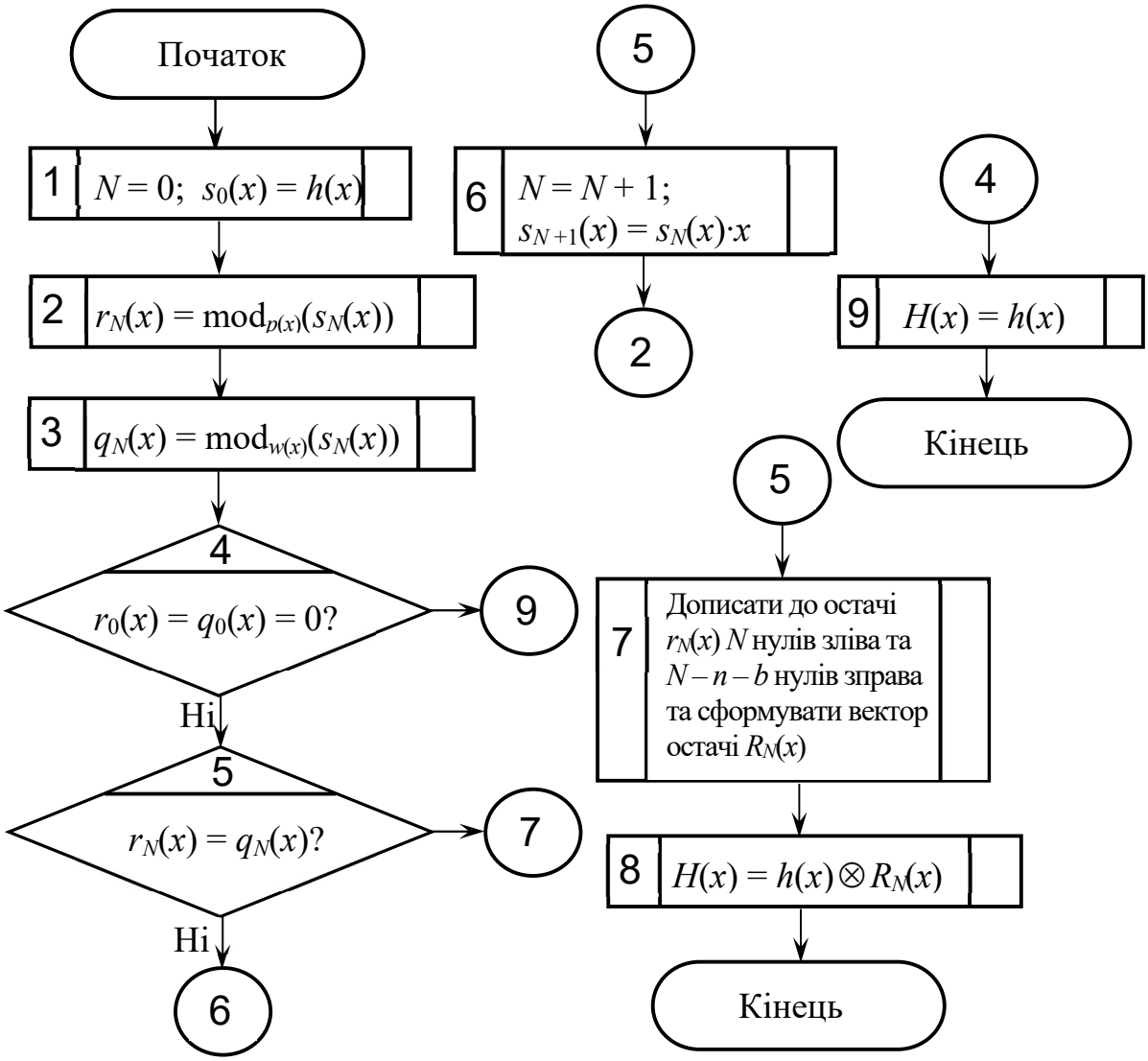


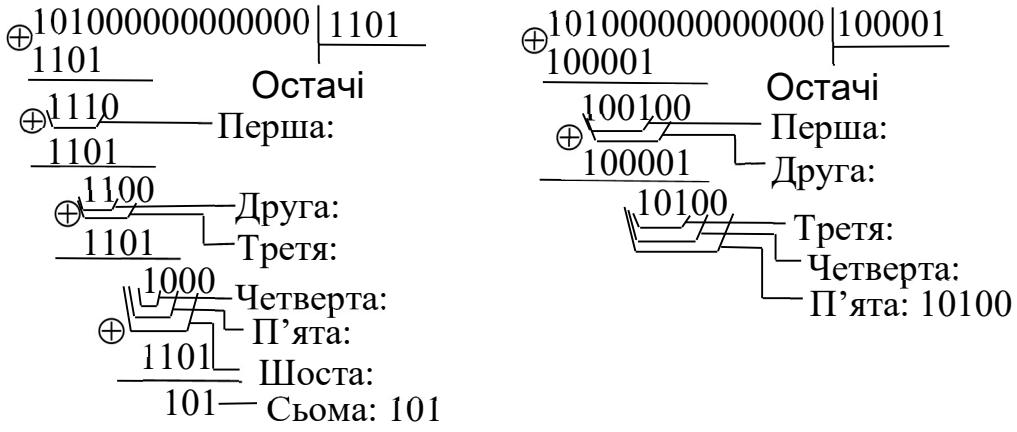
Рис. 3.3 Алгоритм декодування коду Файра та виправлення пакетної помилки

Розглянемо приклад формування коду Файра та пошуку помилки у ньому [5].

Приклад 3.1. Розглянути процес пошуку помилки із синдромом $B(x) = 101$ у коді Файра із твірним поліномом $g(x) = (x^3 + x^2 + 1) \cdot (x^5 + 1)$.

Оскільки твірний поліном містить множник $x^5 + 1$, згідно із співвідношенням (3.17) зрозуміло, що код Файра дозволяє шукати помилки із кратністю $b = (5 + 1)/2 = 3$, а довжина кодової комбінації дорівнює $n = (2^3 - 1) \cdot (2 \cdot 3 - 1) = 35$. В процесі ділення многочлена $B(x) \cdot x^m$ на твірний поліном $x^3 + x^2 + 1$ остача $B(x) = 101$ буде з'являтися на кожному сьомому такті, а в процесі ділення цього многочлена на поліном $x^5 + 1$ остача $B'(x) = 10100$ буде з'являтися на кожному п'ятому такті. Відповідні процеси ділення показані на

рис. 3.4. Зрозуміло, що ділення кодової комбінації на два твірних полінома дає синдром помилки за $T = 7 \cdot 5 = 35$ тактів.



а) б)

Рис. 3.4 Ілюстрація процесу ділення кодової комбінації 101000... на твірні поліноми $x^3 + x^2 + 1$ (а) та $x^5 + 1$ (б)

Це означає, що якщо до синдрому помилки дописати 33 нулі та розділити отриману кодову комбінацію на 1101 та 100001, результатом обох процесів ділення буде визначений синдром помилки. У цьому і полягає алгоритм декодування кодів Файра, наведений на рис 3.3.

Головними параметрами коду Файра є наступні.

1. Тип коду – циклічний несистематичний.

2. Кількість помилок b, які виправляються.

3. Кількість інформаційних розрядів k.

4. Розрядність незвідного твірного поліному m.

5. Розрядність коду $n = k + (2^m - 1) \cdot (2 \cdot b - 1)$. Необхідним є виконання умови, щоб числа $(2^m - 1)$ та $(2 \cdot b - 1)$ були взаємно простими.

6. Надлишковість коду за загальною кількістю розрядів

$$R_n = \frac{(k+m)\cdot(2b-1)-k}{(k+m)\cdot(2b-1)} = \frac{\left((k+\left[\log_2\left((k+1)+\log_2(k+1)\right)\right])\cdot(2b-1)\right)-k}{(k+\left[\log_2\left((k+1)+\log_2(k+1)\right)\right])\cdot(2b-1)} =$$

$$= \frac{(2\cdot b-1)\cdot\left[\log_2\left((k+1)+\log_2(k+1)\right)\right]+2\cdot(b-1)\cdot k}{(k+\left[\log_2\left((k+1)+\log_2(k+1)\right)\right])\cdot(2b-1)}. \qquad (3.19)$$

7. Надлишковість коду за кількістю інформаційних розрядів:

$$R_k = \frac{(k+m)\cdot(2b-1)-k}{k} = \frac{(k+[\log_2((k+1)+\log_2(k+1))])\cdot(2b-1)-k}{k} =$$

$$= \frac{(2\cdot b-1)\cdot[\log_2((k+1)+\log_2(k+1))]+2\cdot(b-1)\cdot k}{k} =$$

$$= \frac{(2\cdot b-1)\cdot[\log_2((k+1)+\log_2(k+1))]}{k} + 2\cdot(b-1). \qquad (3.20)$$

Параметри надлишковості коду Файра розраховуються наступним чином.

1. З використанням співвідношення (2.80) розраховується розрядність s циклічного коду.

2. За умови заданої кількості b помилок, які виправляються, розраховується розрядність коду Файра $n = k + (2^m - 1)\cdot(2\cdot b - 1)$.

3. З використанням співвідношень (2.38), (2.39) розраховуються параметри надлишковості коду Файра. В результаті отримуємо співвідношення (3.19), (3.20).

Розглянемо приклад розрахунку параметрів надлишковості коду Файра.

Приклад 3.2. Розрахувати параметри надлишковості для коду Файра, який кодує восьмирозрядні інформаційні повідомлення та дозволяє виправляти потрійні помилки.

Згідно із співвідношеннями (3.19) та (3.20) маємо:

$$R_n = \frac{(2\cdot b-1)\cdot[\log_2((k+1)+\log_2(k+1))]+2\cdot(b-1)\cdot k}{(k+[\log_2((k+1)+\log_2(k+1))])\cdot(2b-1)} =$$

$$= \frac{(2\cdot 3-1)\cdot[\log_2((8+1)+\log_2(8+1))]+2\cdot(3-1)\cdot 8}{(8+[\log_2((8+1)+\log_2(8+1))])\cdot(2\cdot 3-1)} =$$

$$= \frac{5\cdot[\log_2(9+\log_2(9))]+32}{(8+[\log_2(9+\log_2(9))])\cdot 5} = \frac{5\cdot 4+32}{(8+4)\cdot 5} = 0{,}8667.$$

$$R_k = \frac{(2\cdot b-1)\cdot[\log_2((k+1)+\log_2(k+1))]}{k} + 2\cdot(b-1) =$$

$$= \frac{(2\cdot 3-1)\cdot[\log_2((8+1)+\log_2(8+1))]}{8} + 2\cdot(3-1) =$$

$$= \frac{5 \cdot [\log_2(9 + \log_2(9))]}{8} + 4 = \frac{5 \cdot 4}{8} + 4 = 6{,}5 .$$

Із наведеного прикладу зрозуміло, що головним недоліком кодів Файра є відносно високі коефіцієнти їхньої надлишковості. Іншим недоліком цих кодів є те, що вони є несистематичними, що не дозволяє виділити закодоване слово безпосередньо із кодової комбінації. Декодування послідовностей коду Файра є можливим лише через ділення отриманої кодової комбінації на добуток твірних поліномів $p(x)$ та $(x^{2b-1} + 1)$ [5].

У додатку Л наведена комп'ютерна програма **`Fire`**, яка призначена для формування кодів Файра та декодування їхніх кодових послідовностей, написана мовою програмування системи науково-технічних розрахунків MatLab.

Формування кодів Файра у програмі **`Fire`** здійснюється звичайним чином, через послідовне множення вхідного слова на твірні поліноми. Для декодування послідовностей кодів Файра та пошуку пакетних помилок у програмі реалізований алгоритм, блок-схема якого наведена на рис. 3.3. Для забезпечення роботи програми **`Fire`** викликаються три зовнішні процедури, **`CRC, Bin_Product та Bin_Division`**, коди яких наведені у додатку К. Особливості організації роботи цих процедур та обчислювальні алгоритми, які в них використовуються, були розглянуті у підрозділі 2.5.8.

Програма призначена для формування коду (35, 15) із можливістю виправлення пакетів помилок, кількість яких не перевищує три, та для декодування відповідних кодових послідовностей. За умови внесення незначних змін функціональні можливості програми можуть бути розширені, проте у цьому разі значно збільшується розрядність коду Файра, що ускладнює аналіз результатів розрахунків.

Для реалізації алгоритмів формування кодів Файра та декодування їхніх послідовностей у програмі **`Fire`** використані наступні змінні.

1. Вхідні параметри програми.

`vin` – вектор вхідної послідовності бітів.

nop – тип операції. Можливі значення цього параметру: 1 – формування коду Файра, 2 – декодування кодової послідовності.

Nerr – максимальна кількість помилок у пакеті.

2. Вхідні параметри програми.

vout – результати виконання програми, тобто, комбінація коду Файра за умови **nop=1** або закодоване інформаційне слово за умови **nop=2**.

3. Проміжні змінні.

nv – довжина вхідної послідовності.

TP1 – вектор, який відповідає першому, незвідному твірному поліному.

TP2 – вектор, який відповідає другому твірному поліному ($x^{2b-1}+1$).

RES1M – результат ділення кодового слова на незвідний твірний поліном. Результат записується у вигляді матриці з двома рядками, у першому з яких розташована частка від ділення, а у другому – остача. Для формування матриці ділення використовується функція **Bin_Division**.

RES2M – результат ділення кодового слова на твірний поліном ($x^{2b-1}+1$). Як і для змінної **RES1M**, результат записується у вигляді матриці з двома рядками, у першому з яких розташована частка від ділення, а у другому – остача. Для формування матриці ділення використовується функція **Bin_Division**.

RES1 – остача від ділення кодового слова на незвідний твірний поліном. Формується на основі матриці **RES1M** та являє собою другий рядок цієї матриці.

RES2 – остача від ділення кодового слова на твірний поліном ($x^{2b-1}+1$). Формується на основі матриці **RES2M** та являє собою другий рядок цієї матриці.

RESCOMP1 – остача від ділення кодового слова на незвідний твірний поліном без зайвих лівих нулів.

RESCOMP2 – остача від ділення кодового слова на твірний поліном ($x^{2b-1}+1$) без зайвих лівих нулів.

Слід відзначити, що розмірності векторів **RESCOMP1** та **RESCOMP2** завжди є однаковими та відповідають розмірності вектора помилки. У програмі **Fire** для визначення розмірностей цих векторів використовуються змінні **nrsh1** та **nrsh2** відповідно.

Комп'ютерна реалізація алгоритмів формування коду Файра та декодування його послідовностей у програмі **Fire** є досить простою та не має суттєвих особливостей. Для написання цієї програми були використані засоби програмування системи науково-технічних розрахунків MatLab. Програма **Fire** має модульну структуру, і використання написаних раніше обчислювальних процедур **CRC, Bin_Product та Bin_Division** спрощує організацію програми та розуміння прогркамного коду. Результати тестування програми **Fire** для різних кодових послідовностей також наведені у додатку Л.

3.3 Коди Боуза – Чоудхурі – Хоквінгема

3.3.1 Класифікація кодів Боуза – Чоудхурі – Хоквінгема та особливості їх застосування у сучасних системах зв'язку та інформаційних електронних системах

Серед усіх типів групових кодів сьогодні найбільш поширеними з точки зору використання у сучасних інформаційних системах є коди Боуза – Чоудхурі – Хоквінгема, або, коротко, коди БЧХ (англійський термін – Bose – Chaudhuri – Hocquenghem codes, BCH codes). Розрізняють два типи кодів БЧХ – коди із виявленням помилок та із виправленням помилок. Зазвичай для описання коректувальної здатності кодів БЧХ у технічній літературі використовують наступні абревіатури англійських слів [56, 57].

SED – Single Error Detection, виявлення одиночної помилки.

SEC – Single Error Correction, виправлення одиночної помилки.

DED – Double Error Detection, виявлення подвійної помилки.

DEC – Double Error Correction, виправлення подвійної помилки.

TED – Triple Error Detection, виявлення потрійної помилки.

TEC – Triple Error Correction, виправлення потрійної помилки.

Сьогодні головними галузями застосування кодів БЧХ є наступні стандарти цифрових електронних систем та систем зв'язку [56, 57].

1. Супутникові системи передавання звукових повідомлень.

2. Стандарти цифрового телебачення.

3. Комп'ютерні системи запису інформації на оптичні диски стандартів CD, DVD та Blue Ray.

4. Комп'ютерні системи запису інформації на магнітні диски.

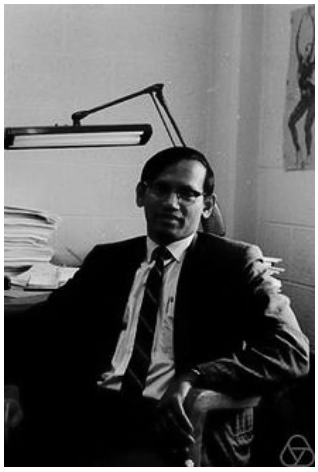
5. Системи криптографічного захисту інформації.

6. Стандарти безпроводового комп'ютерного зв'язку Wi-Fi та WiMAX, зокрема стандартів IEEE 802.3 та IEEE 802.16 [16].

Більш досконале описання сучасних стандартів зв'язку та стандартів інформаційних електронних систем буде наведено у четвертій частині посібника.

Окремим випадком кодів БЧХ є групові коди Ріда – Соломона, які розглядатимуться у підрозділі 3.4.

Рей Чандра Боуз
(1901 – 1987)



Двієнра Кумар
Рей-Чоудхурі
Народився у
1933 р.

Алексіс Хоквінгем
(1908 – 1990)

3.3.2 Параметри кодів Боуза – Чоудхурі – Хоквінгема

Коди БЧХ у теорії кодування зазвичай розглядаються як особлива модифікація циклічних кодів, а спосіб їхньої побудови пов'язаний із методом визначення синдрому багатократної помилки, який був описаний у підрозділі 3.1 [52, 62, 63, 67]. Для групових кодів у технічній літературі часто замість терміну «синдром помилки» використовують термін «локатор помилки» [52, 56, 57]. Головним параметром коду БЧХ є вага вектору помилки t, яку необхідно виправляти.

Припустимо, що m – довільне ціле число, а n – один із дільників числа $q^m - 1$, де q – основа системи числення. У будь-якому полі Галуа $GF(q^m)$ завжди існують елементи n. Будемо вважати, що одним із таких елементів є елемент β. Згідно із теорією циклотомічних класів, описаною у підрозділі 3.6.10 другої частини посібника, всі елементи β^i, за умови $0 \le i < n$, є різними. У теорії кодування існує наступна теорема, яка встановлює зв'язок між кількістю корнів твірного поліному $g(x)$ та коректувальною здатністю циклічного коду.

Теорема 3.1. Якщо твірний поліном циклічного коду $g(x)$ має корені β^{l+1}, β^{l+2}, …, β^{l+r} за умови $l = 0 \le i < n$, тоді мінімальна кодова відстань цього коду є не меншою, ніж $r + 1$, тобто:

$$\exists\left(\left(l = (0 \le i < n)\right), \left(\left(g\left(\beta^{l+1}\right) = 0\right), \left(g\left(\beta^{l+2}\right) = 0\right), \ldots,\right.\right.$$
$$\left.\left.\left(g\left(\beta^{l+2}\right) = 0\right), \ldots, \left(g\left(\beta^{l+r}\right) = 0\right)\right)\right) \Rightarrow d_{\min} \ge r + 1. \qquad (3.21)$$

Для запису логічного виразу (3.21) використана мова теорії предикатів, яка була розглянута у підрозділі 7.1 другої частини посібника.

Мінімальна кодова відстань $d_{\min}$, яка визначається теоремою 3.1 та співвідношенням (3.21), у теорії кодування називається нижньою границею кодів БЧХ.

Для визначення кількості помилок, які виправляються кодом БЧХ, використовується наступна теорема теорії кодування.

Теорема 3.2. Код БЧХ гарантовано виправляє пакет помилок із кількістю t за умови $r = 2 \cdot t$.

Наслідком теорем 3.1 та 3.2 є залежність між мінімальною кодовою відстанню та кількістю помилок, які виправляються:

$$d_{\min} \geq 2t + 1. \tag{3.22}$$

Надамо відповідне визначення.

Визначення 3.1. Мінімальну кодову відстань, яка визначається співвідношеннями (3.21), (3.22), називають конструктивною відстанню кодів БЧХ (англійський термін – designed distance).

Реальна кодова відстань для кодів БЧХ може бути більшою величиною, ніж конструктивна відстань.

Інша важлива теорема теорії кодування пов'язана із визначенням кількості перевірочних символів кодів БЧХ та формулюється наступним чином.

Теорема 3.3. Кількість перевірочних символів коду БЧХ не перевищує величини

$$r_{\max} \leq 2 \cdot m \cdot t. \tag{3.23}$$

За таких умов максимальні коефіцієнти надлишковості кодів БЧХ обчислюються як:

$$R_{n\max} = \frac{2 \cdot t}{k + 2 \cdot t}, \; R_{k\max} = \frac{2 \cdot t}{k}. \tag{3.24}$$

Слід відзначити, що надлишковість кодів БЧХ є мінімальною, що обґрунтовано відповідними теоремами теорії завадостійкого кодування [52, 63]. Оцінки параметрів надлишковості кодів Ріда – Соломона, як різновидів кодів БЧХ, для різних значень імовірності бітової помилки будуть проведені у підрозділі 3.4.12.

3.3.3 Алгоритм формування несистематичних кодів Боуза – Чоудхурі – Хоквінгема та відповідні приклади

Узагальнений алгоритм побудови кодів БЧХ із кодовою відстанню, не меншою за задану, можна записати наступним чином.

1. Визначається, парною чи непарною є кодова відстань для коду, який формується.

2. Для парних значень кодової відстані коренями твірного поліному $g(x)$ є значення 1, β, β^2, ..., β^{2t}.

3. Для непарних значень кодової відстані коренями твірного поліному $g(x)$ є значення β, β^2, ..., β^{2t}.

Тобто, у математичній формі породжувальний поліном можна записати наступним чином:

$$g(x)=\begin{cases}\text{НСК}\left[m_1(x),m_3(x),\ldots,m_{2t-1}(x)\right], & \text{якщо } d_{\min} \text{ є непарним;}\\ \text{НСК}\left[m_0(x),m_1(x),m_3(x),\ldots,m_{2t-1}(x)\right], & \text{якщо } d_{\min} \text{ є парним.}\end{cases} \quad (3.25)$$

4. Перевіряється, чи ділиться поліном $g(x)$ на поліном $m_{2t-1}(x)$. Якщо ділиться – $g(x)$ є твірним поліномом для коду БЧХ, якщо ні – степінь поліному збільшується на 1.

Наприклад, β^2, β^4 є коренями поліному $m_1(x)$, β^6 – коренем поліному $m_6(x)$, β^{10} – коренем поліному $m_5(x)$.

Узагальнений алгоритм побудови кодів БЧХ наведений на рис. 3.5, а способи визначення твірних поліномів для кодів БЧХ із різною коректувальною здатністю розглядатимуться у підрозділі 3.3.5.

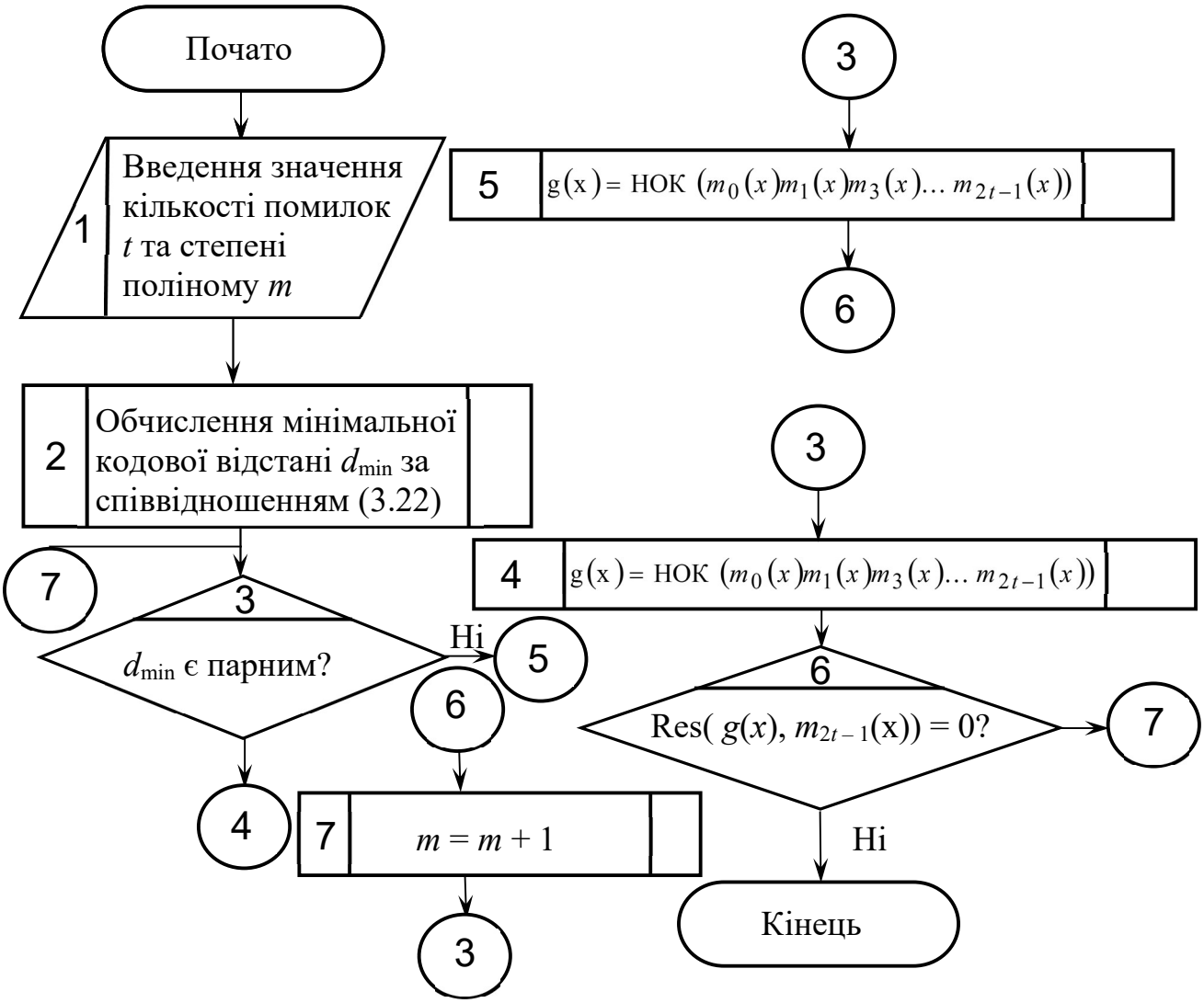


Рис. 3.5 Узагальнений алгоритм побудови несистематичних кодів БЧХ

В цілому способи побудови кодів БЧХ базуються над алгебраїчними операціями над полями Галуа у вигляді поліномів, які були описані у розділі 3

другої частини посібника. Наприклад, різні способи подання поля *GF*(16) як розширеного поля над полем *GF*(2) наведені у таблиці 3.1 [63]. Поле *GF*(16) формується через твірний поліном $p(z) = z^4 + z + 1$. У таблиці 3.1 елементи поля Галуа *GF*(16) записані через степені примітивного елемента, через поліноми над полем *GF*(16), через мінімальні поліноми над полем *GF*(2), а також у десятковій та у двійковій формі. Із даних, наведених у таблиці 3.1, видно, що мінімальні поліноми для різних степенів примітивного елемента α можуть повторюватись. Це пов'язано із теорією циклотомічних класів, описаній у підрозділі 3.6.10 другої частини посібника, а також із тим, що остачі від ділення кодової комбінації на твірний поліном можуть повторюватись, оскільки функція остачі не є гомоморфною [48].

Таблиця 3.1 – Подання поля Галуа *GF*(16) через поля Галуа *GF*(2)

Через степінь примітивного елемента α	Через поліном $s(z)$	У двійковому коді	У десятковому коді	Через мінімальні поліноми $r(x)$
0	0	0000	0	–
α^0	1	0001	1	$x + 1$
α^1	z	0010	2	$x^4 + x + 1$
α^2	z^2	0100	4	$x^4 + x + 1$
α^3	z^3	1000	8	$x^4+x^3+x^2+x+1$
α^4	$z + 1$	0011	3	$x^4 + x + 1$
α^5	$z^2 + z$	0110	6	$x^2 + x + 1$
α^6	$z^3 + z^2$	1100	12	$x^4+x^3+x^2+x+1$
α^7	$z^3 + z + 1$	1011	11	$x^4 + x^3 + 1$
α^8	$z^2 + 1$	0101	5	$x^4 + x + 1$
α^9	$z^3 + z$	1010	10	$x^4+x^3+x^2+x+1$
α^{10}	$z^2 + z + 1$	0111	7	$x^2 + x + 1$
α^{11}	$z^3 + z^2 + z$	1110	14	$x^4 + x^3 + 1$
α^{12}	z^3+z^2+z+1	1111	15	$x^4+x^3+x^2+x+1$
α^{13}	$z^3 + z^2 + 1$	1101	13	$x^4 + x^3 + 1$
α^{14}	$z^3 + 1$	1001	9	$x^4 + x^3 + 1$

Розглянемо типові приклади формування твірних поліномів для кодів БЧХ з використанням співвідношення (3.25) та алгоритму, наведеного на рис. 3.5.

Приклад 3.3. Побудувати твірний поліном для коду БЧХ із довжиною 15 символів, який виправляє подвійні помилки.

Згідно із формулою (3.25), твірний поліном для такого коду формується через пошук найбільшого спільного кратного НСК $[f_1(x), f_2(x), f_3(x), f_1(x)]$ для незвідних поліномів четвертого порядку. Відповідно, згідно із таблицями незвідних поліномів, наведеними у додатку І та у таблиці 3.1, можна записати:

$$g(x) = \text{НСК}\,[f_1(x), f_2(x), f_3(x), f_1(x)] = \text{НСК}\,[x^4 + x + 1, x^4 + x + 1,$$
$$x^4 + x^3 + x^2 + x + 1, x^4 + x + 1] = (x^4 + x + 1) \cdot (x^4 + x^3 + x^2 + x + 1) =$$
$$= x^8 + x^7 + x^6 + x^4 + 1. \qquad (3.26)$$

Із таблиці 3.1 зрозуміло, що функції $f_j(x)$ для твірного поліному, який сформований через співвідношення (3.26), відповідають степеням α^j примітивного елемента α.

Тобто, маємо твірний поліном восьмого порядку. Оскільки довжина коду БЧХ n становить 15 символів, кількість інформаційних розрядів $k = 15 - 8 = 7$. Таким чином, параметри сформованого коду БЧХ (15, 7).

Приклад 3.4. Побудувати код БЧХ для послідовності 1010111 з використанням твірного поліному (3.26), який був отриманий в прикладі 3.3.

Код БЧХ формується як результат множення кодової інформаційного слова $d(x)$ на твірний поліном $g(x)$. Враховуючи, що

$$d(x) = 1010111 = x^6 + x^4 + x^2 + x + 1,$$

остаточно маємо:

$$c(x) = d(x) \cdot g(x) = (x^6 + x^4 + x^2 + x + 1) \cdot (x^8 + x^7 + x^6 + x^4 + 1) = x^{14} + x^{13} +$$
$$+ x^{12} + x^{10} + x^{12} + x^6 + x^{12} + x^{11} + x^{10} + x^8 + x^4 + x^{10} + x^9 + x^8 + x^6 + x^2 +$$
$$+ x^9 + x^8 + x^7 + + x^5 + x + x^8 + x^7 + x^6 + x^4 + 1 = x^{14} + x^{13} + x^{12} + x^{11} +$$
$$+ x^{10} + x^8 + x^6 + x^5 + x^2 + x + 1 = 1111101011100111.$$

Приклад 3.5. Побудувати твірний поліном для коду БЧХ із довжиною 15 символів, який виправляє потрійні помилки.

Для даного випадку, згідно із таблицею 3.1, можна записати:

$$g(x) = \text{НСК}\,[f_1(x), f_2(x), f_3(x), f_4(x), f_5(x), f_6(x)] = \text{НСК}\,[x^4 + x + 1, x^4 + x + 1,$$
$$x^4 + x^3 + x^2 + x + 1, x^4 + x + 1, x^2 + x + 1, x^4 + x^3 + x^2 + x + 1] =$$

$$= (x^4 + x + 1) \cdot (x^4 + x^3 + x^2 + x + 1) \cdot (x^2 + x + 1) =$$
$$= (x^8 + x^7 + x^6 + x^4 + 1) \cdot (x^2 + x + 1) = x^{10} + x^9 + x^8 + x^6 + x^2 +$$
$$+ x^9 + x^8 + x^7 + x^5 + x + x^8 + x^7 + x^6 + x^4 + 1 = x^{10} + x^8 + x^5 +$$
$$+ x^4 + x^2 + x + 1. \qquad (3.27)$$

Враховуючи те, що твірний поліном має порядок m = 10, для даного випадку параметри коду БЧХ становлять (15, 5).

Приклад 3.6. Побудувати код БЧХ для послідовності 10101 з використанням твірного поліному (3.27), який був отриманий в прикладі 3.5.

Враховуючи те, що кодове слово 10101 у поліноміальній формі записується як

$$d(x) = x^4 + x^2 + 1,$$

код БЧХ формується як результат множення кодової комбінації $d(x)$ на твірний поліном $g(x)$, який визначається співвідношенням (3.27). Остаточно маємо:

$$c(x) = d(x) \cdot g(x) = (x^4 + x^2 + 1) \cdot (x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1) =$$
$$= x^{14} + x^{12} + x^9 + x^8 + x^6 + x^5 + x^4 + x^{12} + x^{10} + x^7 + x^6 + x^4 + x^3 + x^2 +$$
$$+ x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1 = x^{14} + x^9 + x^7 + x^4 + x^3 + x + 1 =$$
$$= 100001010011011.$$

Можна формувати коди БЧХ і для багатопозиційних цифрових кодових комбінацій, якщо порядок поля Галуа n є складеним. У цьому випадку можливим є розкладання поля Галуа $GF(n)$ поле $GF(m)$, де m – дільник числа n.

Наприклад, у таблиці 3.2 наведені результати розкладання поля Галуа $GF(16)$ через елементи поля $GF(4)$ [63]. Результати сумування та множення для елементів поля Галуа $GF(4)$ записуються наступним чином [63]:

$$0+0=0;\ 0+1=1;\ 0+2=2;\ 0+3=3;\ 1+0=1;\ 1+1=0;\ 1+2=3;\ 1+3=2;$$
$$2+0=2;\ 2+1=3;\ 2+2=0;\ 2+3=1;\ 3+0=3;\ 3+1=2;\ 3+2=1;\ 3+3=0;$$
$$0\cdot 0=0;\ 0\cdot 1=0;\ 0\cdot 2=0;\ 0\cdot 3=0;\ 1\cdot 0=0;\ 1\cdot 1=1;\ 1\cdot 2=2;\ 1\cdot 3=3; \qquad (3.28)$$
$$2\cdot 0=0;\ 2\cdot 1=2;\ 2\cdot 2=3;\ 2\cdot 3=1;\ 3\cdot 0=0;\ 3\cdot 1=3;\ 3\cdot 2=1;\ 3\cdot 3=2.$$

З використанням мінімальних поліномів, наведених у таблиці 3.2, можна формувати твірні поліноми для кодів БЧХ. Розглянемо відповідні приклади [63].

Таблиця 3.2 – Подання поля Галуа *GF*(16) через поля Галуа *GF*(4)

Через степінь примітивного елемента α	Через поліном $s(z)$	У четверковому коді	У десятковому коді	Через мінімальні поліноми $r(x)$
0	0	00	0	–
α^0	1	01	1	$x + 1$
α^1	z	10	4	$x^2 + x + 2$
α^2	$z + 2$	12	6	$x^2 + x + 3$
α^3	$3z + 2$	32	14	$x^2 + 3x + 1$
α^4	$z + 1$	11	5	$x^2 + x + 2$
α^5	2	02	2	$x + 2$
α^6	$2z$	20	8	$x^2 + 2x + 1$
α^7	$2z + 3$	23	11	$x^2 + 2x + 2$
α^8	$z + 3$	13	7	$x^2 + x + 3$
α^9	$2z + 2$	22	10	$x^2 + 2x + 1$
α^{10}	3	03	3	$x + 3$
α^{11}	$3z$	30	12	$x^2 + 3x + 3$
α^{12}	$3z + 1$	31	13	$x^2 + 3x + 1$
α^{13}	$2z + 1$	21	9	$x^2 + 2x + 2$
α^{14}	$3z + 3$	33	15	$x^2 + 3x + 3$

Приклад 3.7. Побудувати твірний поліном для коду БЧХ над полем Галуа *GF*(4), який виправляє одиночні помилки.

Згідно із співвідношенням (3.25) можна записати:

$$g(x) = \text{НСК}\,[f_1(x), f_2(x)]. \qquad (3.29)$$

Тоді, враховуючі поліноміальне подання, яке наведено у таблиці 3.2, співвідношення (3.29), з урахуванням алгебраїчних операцій (3.28), заданих у полі *GF*(4) переписується наступним чином:

$$g(x) = \text{НСК}\,[f_1(x), f_2(x)] = (x^2 + x + 2) \cdot (x^2 + x + 3) = x^4 + x^3 + 3x^2 + x^3 + x^2 +$$

$$+ 3x + 2x^2 + 2x + 1 = x^4 + (1 + 1) \cdot x^3 + (3 + 1 + 2) \cdot x^2 + (3 + 2) \cdot x + 1 =$$
$$= x^4 + (2 + 2) \cdot x^2 + (3 + 2) \cdot x + 1 = x^4 + x + 1. \quad (3.30)$$

В результаті отриманий твірний поліном $g(x)$ для коду БЧХ (15, 11) над полем Галуа $GF(4)$, який виправляє одиночні помилки. У даному випадку параметри (15, 11) означають, що кодуються 11 двобітових послідовностей, тобто 22 біти, а результуючий код БЧХ має $15 \cdot 2 = 30$ бітів.

Формування багатопозиційних кодів БЧХ також здійснюється через множення поліномів. Необхідно виконати стандартну поліноміальну алгебраїчну операцію:

$$c(x) = d(x) \cdot g(x), \quad (3.31)$$

де $c(x)$ – комбінація коду БЧХ, $d(x)$ – інформаційне слово, $g(x)$ – твірний поліном. Особливість виконання цієї операції полягає у тому, що сумування коефіцієнтів при однакових степенях змінної x здійснюється з використанням алгебраїчних операцій над елементами поля Галуа меншого порядку, аналогічних співвідношенням (3.28).

Розглянемо відповідний приклад.

Приклад 3.8. Побудувати над полем Галуа $GF(4)$ код БЧХ, який виправляє одиночні помилки, для інформаційного слова 23113020321.

Згідно із співвідношенням (3.31), будемо шукати код БЧХ $c(x)$ як добуток визначеного інформаційного слова $d(x)$ = 23113020321 на твірний поліном (3.30) $g(x) = x^4 + x + 1$, визначений у прикладі 3.7. Під час множення поліномів для зведення подібних доданків будемо використовувати співвідношення (3.28) над елементами поля Галуа $GF(4)$. Відповідно маємо:

$$c(x) = d(x) \cdot g(x) = (2 \cdot x^{10} + 3 \cdot x^9 + x^8 + x^7 + 3 \cdot x^6 + 2 \cdot x^4 + 3 \cdot x^2 + 2 \cdot x + 1) \cdot (x^4 +$$
$$+ x + 1) = 2 \cdot x^{14} + 3 \cdot x^{13} + x^{12} + x^{11} + 3 \cdot x^{10} + 2 \cdot x^8 + 3 \cdot x^6 + 2 \cdot x^5 + x^4 +$$
$$+ 2 \cdot x^{11} + 3 \cdot x^{10} + x^9 + x^8 + 3 \cdot x^7 + 2 \cdot x^5 + 3 \cdot x^3 + 2 \cdot x^2 + x +$$
$$+ 2 \cdot x^{10} + 3 \cdot x^9 + x^8 + x^7 + 3 \cdot x^6 + 2 \cdot x^4 + 3 \cdot x^2 + 2 \cdot x + 1 =$$
$$= 2 \cdot x^{14} + 3 \cdot x^{13} + x^{12} + (1 + 2) \cdot x^{11} + (3 + 3 + 2) \cdot x^{10} + (3 + 1) \cdot x^9 +$$
$$+ (2 + 1 + 1) \cdot x^8 + (3 + 1) \cdot x^7 + (3 + 3) \cdot x^6 + (2 + 2) \cdot x^5 + (1 + 2) \cdot x^4 +$$
$$+ 3 \cdot x^3 + (2 + 3) \cdot x^2 + (1 + 2) \cdot x + 1 = 2 \cdot x^{14} + 3 \cdot x^{13} + x^{12} + 3 \cdot x^{11} + 2 \cdot x^{10} +$$

$$+ 2\cdot x^9 + 2\cdot x^8 + 2\cdot x^7 + 0\cdot x^6 + 0\cdot x^5 + 3\cdot x^4 + 3\cdot x^3 + x^2 + 3\cdot x + 1 =$$

$$= 231322220033131.$$

Тобто, п'ятнадцятирозрядний код БЧХ для інформаційного слова з 11 символів $d(x) = 23113020321$ становить $c(x) = 231322220033131$. Коректувальна здатність такого коду полягає у можливості виправлення будь-якої одиночної помилки.

Приклад 3.9. Побудувати твірний поліном для коду БЧХ над полем Галуа $GF(4)$, який виправляє подвійні помилки.

Для цього прикладу, згідно із співвідношенням (3.25), маємо:

$$g(x) = \text{НСК}\,[f_1(x), f_2(x), f_3(x), f_4(x)]. \qquad (3.32)$$

Використовуючи мінімальні поліноми, які записані у таблиці 3.2, перепишемо співвідношення (3.32) наступним чином:

$$g(x) = \text{НСК}\,[(x^2 + x + 2), (x^2 + x + 3), (x^2 + 3\cdot x + 1), (x^2 + x + 2)] =$$

$$= (x^2 + x + 2)\cdot(x^2 + x + 3)\cdot(x^2 + 3\cdot x + 1) = (x^4 + x^3 + 2\cdot x^2 + x^3 + x^2 +$$

$$+ 2\cdot x + 3\cdot x^2 + 3\cdot x + 3\cdot 2\,)\cdot(x^2 + 3\cdot x + 1) = (x^4 + (1 + 1)\cdot x^3 + (2 + 1 + +$$

$$3)\cdot x^2 + (2 + 3)\cdot x + 1)\cdot(x^2 + 3\cdot x + 1) = (x^4 + (3 + 3)\cdot x^2 + 1\cdot x +$$

$$+ 1)\cdot(x^2 + 3\cdot x + 1) = (x^4 + x + 1)\cdot(x^2 + 3\cdot x + 1) = x^6 + x^3 + x^2 + 3\cdot x^5 +$$

$$+ 3\cdot x^2 + 3x + x^4 + x + 1 = x^6 + 3\cdot x^5 + x^4 + x^3 + (1 + 3)\cdot x^2 + (1 + 3)\cdot x +$$

$$+ 1 = x^6 + 3\cdot x^5 + x^4 + x^3 + 2\cdot x^2 + 2\cdot x + 1.$$

В результаті виконання цього завдання отриманий твірний поліном

$$g(x) = x^6 + 3\cdot x^5 + x^4 + x^3 + 2\cdot x^2 + 2\cdot x + 1 \qquad (3.33)$$

для коду БЧХ (15, 9) над полем Галуа $GF(4)$, який виправляє подвійні помилки. Наведемо приклад формування такого коду.

Приклад 3.10. Побудувати над полем Галуа $GF(4)$ код БЧХ, який виправляє подвійні помилки, для інформаційного слова 203010021.

Для розв'язування поставленої задачі використаємо співвідношення (3.31) за умови, що твірний поліном $g(x)$ задається співвідношенням (3.33). Відповідно, враховуючи співвідношення (3.28) для алгебраїчних операцій над елементами поля Галуа $GF(4)$, маємо:

$$c(x) = d(x)\cdot g(x) = (2\cdot x^8 + 3\cdot x^6 + x^4 + 2\cdot x + 1)\cdot(x^6 + 3\cdot x^5 + x^4 + x^3 +$$

$$+ 2\cdot x^2 + 2\cdot x + 1) = 2\cdot x^{14} + 3\cdot x^{12} + x^{10} + 2\cdot x^7 + x^6 + (2\cdot 3)\cdot x^{13} +$$
$$+ (3\cdot 3)\cdot x^{11} + 3\cdot x^9 + (2\cdot 3)\cdot x^6 + 3\cdot x^5 + 2\cdot x^{12} + 3\cdot x^{10} + x^8 + 2\cdot x^5 + x^4 +$$
$$+ 2\cdot x^{11} + 3\cdot x^9 + x^7 + 2\cdot x^4 + x^3 + (2\cdot 2)\cdot x^{10} + (3\cdot 2)\cdot x^8 + 2\cdot x^6 + (2\cdot 2)\cdot x^3 +$$
$$+ 2\cdot x^2 + (2\cdot 2)\cdot x^9 + (3\cdot 2)\cdot x^7 + 2\cdot x^5 + (2\cdot 2)\cdot x^2 + 2\cdot x + 2\cdot x^8 + 3\cdot x^6 + x^4 +$$
$$+ 2\cdot x + 1 = 2\cdot x^{14} + 3\cdot x^{12} + x^{10} + 2\cdot x^7 + x^6 + x^{13} + 2\cdot x^{11} + 3\cdot x^9 +$$
$$+ x^6 + 3\cdot x^5 + 2\cdot x^{12} + 3\cdot x^{10} + x^8 + 2\cdot x^5 + x^4 + 2\cdot x^{11} + 3\cdot x^9 + x^7 +$$
$$+ 2\cdot x^4 + x^3 + 3\cdot x^{10} + x^8 + 2\cdot x^6 + 3\cdot x^3 + 2\cdot x^2 + 3\cdot x^9 + x^7 + 2\cdot x^5 +$$
$$+ 3\cdot x^2 + 2\cdot x + 2\cdot x^8 + 3\cdot x^6 + x^4 + 2\cdot x + 1 = 2\cdot x^{14} + x^{13} + (3 + 2)\cdot x^{12} +$$
$$+ (2 + 2)\cdot x^{11} + (1 + 3 + 3)\cdot x^{10} + (3 + 3 + 3)\cdot x^9 + (1 + 1 + 3)\cdot x^8 +$$
$$+ (2 + 1 + 1)\cdot x^7 + (1 + 2 + 3)\cdot x^6 + (3 + 2 + 2)\cdot x^5 + (1 + 2 + 1)\cdot x^4 +$$
$$+ (1 + 3)\cdot x^3 + (2 + 3)\cdot x^2 + (2 + 2)\cdot x + 1 = 2\cdot x^{14} + x^{13} + x^{12} + x^{10} +$$
$$+ 3\cdot x^9 + 3\cdot x^8 + 2\cdot x^7 + (3 + 3)\cdot x^6 + 3\cdot x^5 + 2\cdot x^4 + 2\cdot x^3 + x^2 + 1 =$$
$$= 2\cdot x^{14} + x^{13} + x^{12} + x^{10} + 3\cdot x^9 + 3\cdot x^8 + 2\cdot x^7 + x^5 + 2\cdot x^4 + 2\cdot x^3 + x^2 + 1 =$$
$$= 211013320122101.$$

Тобто, $c(x)$ = 211013320122101. Це п’ятнадцятирозрядний чотирьохпозиційний код БЧХ для дев’ятирозрядного чотирьохпозиційного числа $d(x)$ = 203010021, який виправляє подвійні помилки.

Приклад 3.11. Побудувати твірний поліном для коду БЧХ над полем Галуа $GF(4)$, який виправляє потрійні помилки.

Для цього прикладу, згідно із співвідношенням (3.25), маємо:

$$g(x) = \text{НСК}\,[f_1(x), f_2(x), f_3(x), f_4(x)\,), f_5(x), f_6(x)]. \qquad (3.34)$$

Використовуючи мінімальні поліноми, які записані у таблиці 3.2, перепишемо співвідношення (3.32) наступним чином:

$$g(x) = \text{НСК}\,[(x^2 + x + 2), (x^2 + x + 3), (x^2 + 3\cdot x + 1), (x^2 + x + 2),$$
$$(x + 2), (x^2 + 2\cdot x + 1)] = g_2(x)\cdot(x + 2)\cdot(x^2 + 2\cdot x + 1), \qquad (3.35)$$

де $g_2(x)$ – твірний поліном (3.33), який використовується для формування коду БЧХ (15, 9), що дозволяє виправляти подвійні помилки. Тобто, із співвідношення (3.35) можна зробити висновок, що якщо відомий твірний поліном для створення коду БЧХ із низькою коректувальною здатністю над

заданим полем Галуа, цей поліном може бути використаний для формування коду із більш високою коректувальною здатністю. Ця властивість кодів БЧХ безпосередньо випливає із співвідношення (3.25) та може бути ефективно використана для спрощення алгебраїчних операцій над поліномами [63]. Згідно із співвідношенням (3.35) можна записати:

$$g(x) = g_2(x)\cdot(x+2)\cdot(x^2+2\cdot x+1) = (x^6+3\cdot x^5+x^4+x^3+2\cdot x^2+2\cdot x+1)\cdot(x+$$
$$+2)\cdot(x^2+2\cdot x+1) = (x^6+3\cdot x^5+x^4+x^3+2\cdot x^2+2\cdot x+1)\cdot(x^3+2\cdot x^2+$$
$$+x+2\cdot x^2+(2\cdot 2)\cdot x+1) = (x^6+3\cdot x^5+x^4+x^3+2\cdot x^2+2\cdot x+$$
$$+1)\cdot(x^3+2\cdot x^2+x+2\cdot x^2+3\cdot x+1) = (x^6+3\cdot x^5+x^4+x^3+2\cdot x^2+$$
$$+2\cdot x+1)\cdot(x^3+(2+2)\cdot x^2+(1+3)\cdot x+1) = (x^6+3\cdot x^5+x^4+x^3+$$
$$+2\cdot x^2+2\cdot x+1)\cdot(x^3+2\cdot x+1) = x^9+3\cdot x^8+x^7+x^6+2\cdot x^5+2\cdot x^4+$$
$$+x^3+2\cdot x^7+(3\cdot 2)\cdot x^6+2\cdot x^5+2\cdot x^4+(2\cdot 2)\cdot x^3+(2\cdot 2)\cdot x^2+2\cdot x+$$
$$+x^6+3\cdot x^5+x^4+x^3+2\cdot x^2+2\cdot x+1 = x^9+3\cdot x^8+x^7+x^6+2\cdot x^5+$$
$$+2\cdot x^4+x^3+2\cdot x^7+x^6+2\cdot x^5+2\cdot x^4+3\cdot x^3+3\cdot x^2+2\cdot x+$$
$$+x^6+3\cdot x^5+x^4+x^3+2\cdot x^2+2\cdot x+1 = x^9+3\cdot x^8+(1+2)\cdot x^7+$$
$$+(1+1+1)\cdot x^6+(2+2+3)\cdot x^5+(2+2+1)\cdot x^4+(1+3+1)\cdot x^3+$$
$$+(3+2)\cdot x^2+(2+2)\cdot x+1 = x^9+3\cdot x^8+3\cdot x^7+3\cdot x^5+x^4+3\cdot x^3+x^2+1.$$

В результаті виконання цього завдання отриманий твірний поліном

$$g(x) = x^9+3\cdot x^8+3\cdot x^7+3\cdot x^5+x^4+3\cdot x^3+x^2+1 \qquad (3.36)$$

для коду БЧХ (15, 6) над полем Галуа $GF(4)$, який виправляє потрійні помилки. Наведемо приклад формування такого коду.

Приклад 3.12. Побудувати над полем Галуа $GF(4)$ код БЧХ, який виправляє потрійні помилки, для інформаційного слова 203001.

Використовуючи співвідношення (3.31), а також враховуючи те, що твірний поліном $g(x)$ для коду БЧХ над полем Галуа $GF(4)$, який виправляє потрійні помилки, визначається співвідношенням (3.36), кодове слово можна знайти наступним чином:

$$c(x) = d(x)\cdot g(x) = (2\cdot x^5+3\cdot x^3+1)\cdot(x^9+3\cdot x^8+3\cdot x^7+3\cdot x^5+x^4+3\cdot x^3+x^2+1) =$$
$$= 2\cdot x^{14}+(2\cdot 3)\cdot x^{13}+(2\cdot 3)\cdot x^{12}+(2\cdot 3)\cdot x^{10}+2\cdot x^9+(2\cdot 3)\cdot x^8+2\cdot x^7+2\cdot x^5+$$

$$+3\cdot x^{12}+(3\cdot 3)\cdot x^{11}+(3\cdot 3)\cdot x^{10}+(3\cdot 3)\cdot x^{8}+3\cdot x^{7}+(3\cdot 3)\cdot x^{6}+3\cdot x^{5}+3\cdot x^{3}+$$
$$+x^{9}+3\cdot x^{8}+3\cdot x^{7}+3\cdot x^{5}+x^{4}+3\cdot x^{3}+x^{2}+1=2\cdot x^{14}+x^{13}+x^{12}+x^{10}+$$
$$+2\cdot x^{9}+x^{8}+2\cdot x^{7}+2\cdot x^{5}+3\cdot x^{12}+2\cdot x^{11}+2\cdot x^{10}+2\cdot x^{8}+3\cdot x^{7}+2\cdot x^{6}+$$
$$+3\cdot x^{5}+3\cdot x^{3}+x^{9}+3\cdot x^{8}+3\cdot x^{7}+3\cdot x^{5}+x^{4}+3\cdot x^{3}+x^{2}+1=2\cdot x^{14}+x^{13}+$$
$$+(1+3)\cdot x^{12}+2\cdot x^{11}+(1+2)\cdot x^{10}+(2+1)\cdot x^{9}+(1+3)\cdot x^{8}+(3+3)\cdot x^{7}+$$
$$+2\cdot x^{6}+(3+3)\cdot x^{5}+x^{4}+(3+3)\cdot x^{3}+x^{2}+1=2\cdot x^{14}+x^{13}+2\cdot x^{12}+2\cdot x^{11}+$$
$$+3\cdot x^{10}+3\cdot x^{9}+2\cdot x^{8}+2\cdot x^{6}+x^{4}+x^{2}+1=212233202010101.$$

Тобто, для коду БЧХ над полем Галуа $GF(4)$, який виправляє потрійні помилки, вхідному слову $d(x)$ = 203001 відповідає кодова послідовність $c(x)$ = 212233202010101.

Зрозуміло, що розглянуті вище коди БЧХ є різновидом несистематичних циклічних кодів, способи побудови яких розглядалися у підрозділі 2.5.2. Описаний спосіб побудови несистематичних кодів БЧХ є досить простим, тому тепер головним завданням є пошук ефективних алгоритмів їхнього декодування, які розглядатимуться у наступному підрозділі. Надалі будуть розглянуті способи формування систематичних кодів БЧХ, а також відповідні кодувальні та декодувальні цифрові електронні пристрої. У підрозділі 3.4 буде розглянутий код Ріда – Соломона як один із різновидів багатопозиційних кодів БЧХ.

Що стосуєтсья звичайних кодів БЧХ, у сучасній кодувальній електронній апаратурі здебільшого використовуються систематичні двійкові коди, які виправляють подвійні та потрійні помилки. Апаратні та програмні засоби для формування таких кодів та декодування їхніх послідовностей розглядатимуться у підрозділах 3.3.6 та 3.3.7. Перевагою апаратних засобів формування кодів БЧХ та декодуваня їхніх послідовностей є більш висока швидкодія, а перевагою програмних засобів – більша гнучкість та можливість підбору найбільш ефективних кодів для заданих параметрів системи. Тому сьогодні широке впровадження в кодувальних системах знаходять програмовані логічні інтегральні схеми (ПЛІС), загальні принципи роботи яких були описані у шостому розділі першої частини посібника [1].

3.3.4 Алгоритми декодування кодів Боуза – Чоудхурі – Хоквінгема

3.3.4.1 Узагальнене описання алгоритмів декодування кодів Боуза – Чоудхурі – Хоквінгема та їхня класифікація

Існують різні способи декодування кодів БЧХ, вибір яких, насамперед, визначається коректувальною здатністю коду та загальною кількістю його розрядів n. Також розрізняють числові алгоритми декодування кодів БЧХ, які можуть бути використані в програмованих електронних пристроях, та логічно-обчислювальні алгоритми, які зазвичай застосовуються в кодувальній цифровій електронній апаратурі без засобів програмування, або в сучасній електронній апаратурі на програмованих логічних інтегральних схемах (ПЛІС) [1, 52, 56, 57, 62, 63, 78, 79].

Оскільки, з точки зору загальної класифікації, коди БЧХ є одним із різновидів циклічних кодів, для них можуть бути застосовані будь-які способи та алгоритми декодування циклічних кодів, які розглядалися у підрозділі 2.5. Проте сьогодні розроблені значно більш ефективні алгоритми декодування групових кодів, тому для декодування кодів БЧХ зазвичай стандартні алгоритми обробки циклічних кодів не використовуються.

Зазвичай для декодування групових кодів, зокрема кодів БЧХ, використовують різні ітераційні алгоритми, які можна класифікувати наступним чином [52, 62, 63].

1. Прямий розв’язок системи лінійних рівнянь з використанням матричних перетворень та алгебраїчних операцій над примітивними елементами, відомими із теорії груп. Відповідні операції були описані у другому та третьому розділі другої частини посібника. Цей метод називається методом Пітерсона – Горенстейна – Цирлера (ПГЦ). Вперше запропонований американським математиком Пітерсоном для двійкових кодів ($q = 2$), цей метод пізніше був розвинений для багатопозиційних кодів у роботах Горенстейна та Цирлера [52, 62, 63]. Недоліком алгоритму ПГЦ є те, що складність розв’язування систем лінійних рівнянь для цього алгоритму зростає пропорційно кубу мінімальної кодової відстані $d_{\min}$. З цієї причини алгоритм

ПГЦ зазвичай використовується на практиці лише за умови низького значення $d_{\min}$. Це, насамперед, пов'язано з тим, що алгоритм ПГЦ передбачає обернення матриць розмірності t x t, а ця процедура в обчислювальній математиці, як відомо, є досить ресурсоємною [11, 13, 14].

Проте історично саме алгоритм ПГЦ був розроблений першим, і його найважливішою істотною перевагою є простота та наочність. На основі цього алгоритму можна найкращим чином зрозуміти сутність ідеї виправлення багаторазових помилок у кодах БЧХ.

2. Алгоритм Берлекемпа – Мессі (АМБ) (англійський термін – Berleckamp – Massey Algorithm, BMA). Перевагою цього алгоритму є те, що в ньому для обчислення позицій помилок замість процедури обернення матриць використовується перетворення поліномів над полями Галуа, які базуються на відомій теоремі Форні [52, 62, 63]. З точки зору обчислювальної складності та розуміння алгоритм Берлекемпа – Мессі є більш складним, ніж алгоритм ПГЦ, оскільки він містить більшу кількість логічних операцій, перевірок та розгалужень. Проте для великих значень параметра $d_{\min}$ обчислення за АМБ зазвичай проводяться за меншу кількість ітерацій. Крім цього, відомо, що операції порівняння, яких досить багато в АМБ, є достатньо простими для електронних обчислювальних пристроїв та легко ними виконуються на апаратному рівні [11, 13, 14]. Значно складніше в електронній обчислювальній апаратурі реалізовувати алгебраїчні операції над натуральними числами, оскільки під час виконання таких операцій може бути втрачена обчислювальна точність [11, 13, 14].

3. Алгоритм Евкліда, який загалом базується на пошуку найбільшого спільного кратного двох поліномів.

4. Процедура Ченя, за допомогою якої можна визначити позиції помилок. Ця процедура пов'язана із обчисленням коренів поліномів над полем Галуа та легко реалізується в декодерах кодів БЧХ на апаратному рівні з використанням логічних електронних схем. Також на основі процедури Ченя побудований алгоритм ПГЦ.

Узагальнена структурна схема декодера кодів БЧХ наведена на рис. 3.6.

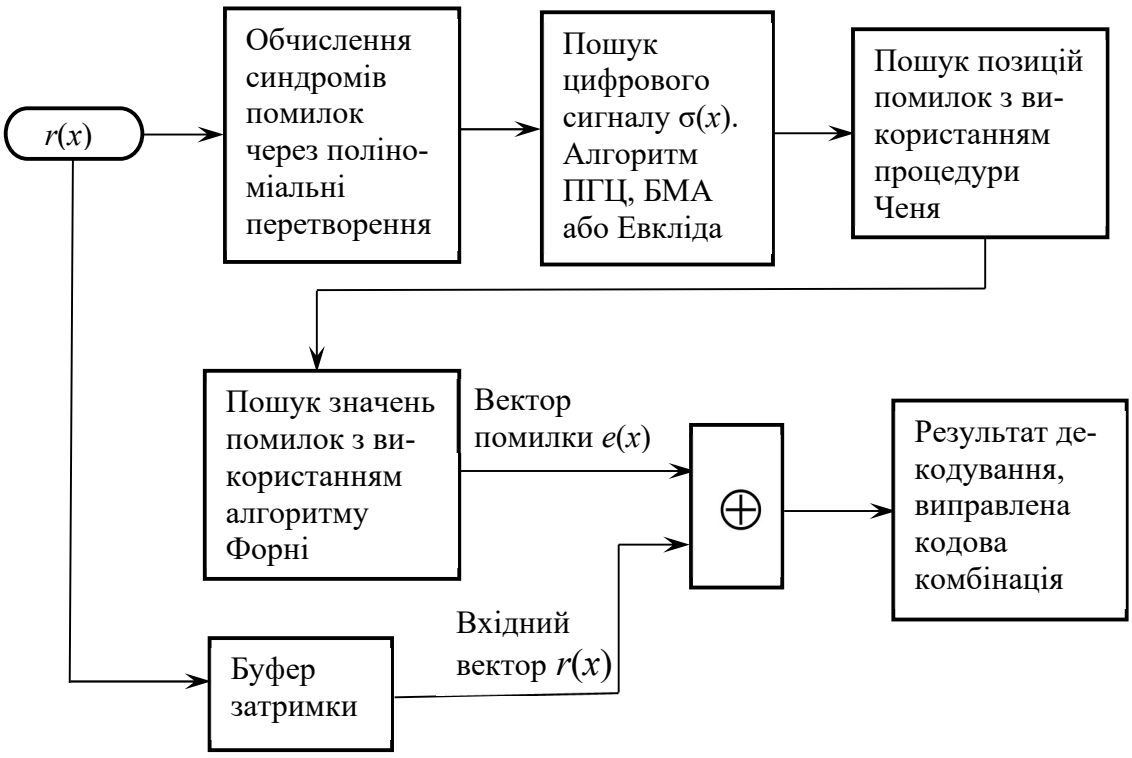


Рис. 3.6 Структурна схема декодера кодів БЧХ

Узагальнений порівняльний аналіз розглянутих вище алгоритмів буде наведений у наступних підрозділах.

3.3.4.2 Алгоритм Пітерсона – Горенстейна – Цирлера та процедура Ченя

Для описання алгоритму ПГЦ розглянемо довільний код БЧХ, в основі якого лежить елемент поля Галуа α, який може бути не примітивним. На початковому етапі аналізу розглянемо вектор помилки, який запишемо у вигляді полінома [52, 62, 63]:

$$e(x) = e_{n-1}x^{n-1} + e_{n-2}x^{n-2} + \ldots + e_1 x + e_0 , \qquad (3.37)$$

у якому не більше, ніж t коефіцієнтів відрізняються від нуля. Будемо вважати, що кількість помилок дорівнює ν, де $0 \le \nu \le t$. Зрозуміло, що цим помилкам відповідають невідомі номера позицій i_1, i_2, …, i_ν. Тоді поліном (3.37) можна переписати у спрощеному вигляді, без нульових коефіцієнтів:

$$e(x) = e_{i_1}x^{i_1} + e_{i_2}x^{i_2} + \ldots + e_{i_\nu}x^{i_\nu} , \qquad (3.38)$$

де e_{i_l} – величина, яка описує характер помилки. Для двійкових кодів БЧХ необхідність розв'язування задачі пошуку величини помилки відпадає, оскільки у разі наявності помилки завжди виконується тотожність $e_{i_l} = 1$. На даному етапі, який можна назвати етапом постановки задачі декодування, у лінійному рівнянні (3.38) існує велика кількість невідомих параметрів, а саме:

1. вектори e_{i_l};

2. положення помилок $i_1, i_2, \ldots, i_\nu$;

3. загальна кількість помилок ν.

Для визначення положення помилок необхідно знайти всі ці величини. Будемо шукати компоненту синдрому помилок S_1 як значення полінома (3.38) за умови $x = \alpha$. Відповідно, маємо [52, 62, 63]:

$$S_1 = v(\alpha) = c(\alpha) + e(\alpha) = e(\alpha) = e_{i_1}\alpha^{i_1} + e_{i_2}\alpha^{i_2} + \ldots + e_{i_\nu}\alpha^{i_\nu}. \quad (3.39)$$

Отриманий вираз (3.39) можна значно спростити через введення відповідних позначень змінних. Надалі будемо вважати, що

$$Y_l = e_{i_l},\ X_l = \alpha^{i_l}, \quad (3.40)$$

де Y_l – величина помилки, X_l – локатор помилки, який описується через елемент поля Галуа, i_l – положення помилки із номером l. Для коректного розв'язування рівняння (3.39) необхідно, щоб всі локатори помилок X_l для будь-якого значення l були унікальними. Ця умова забезпечується у разі, якщо порядок елементу α дорівнює n та відповідає періоду циклотомічного класу цього елементу.

Згідно із введеними позначеннями змінних (3.40), перепишемо рівняння (3.39) у вигляді [52, 62, 63]:

$$S_1 = Y_1 \cdot X_1 + Y_2 \cdot X_2 + \ldots + Y_\nu \cdot X_\nu. \quad (3.41)$$

Аналогічно формулі (3.39), запишемо значення многочлена локатора помилок S_j для всіх степенів елементу α, які входять в розкладання твірного

поліному $g(x)$. У загальному, формалізованому вигляді, поліном синдромів помилок S_j для значень $j = 1, \ldots, 2 \cdot t$ записується наступним чином [52, 62, 63]:

$$S_j = v\left(\alpha^j\right) = c\left(\alpha^j\right) + e\left(\alpha^j\right) = e\left(\alpha^j\right). \tag{3.42}$$

Таким чином, враховуючи (3.40), отримуємо систему нелінійних степеневих рівнянь, аналогічну системі (3.7) [52, 62, 63]:

$$\begin{aligned} & S_1 = Y_1 \cdot X_1 + Y_2 \cdot X_2 + \ldots + Y_\nu \cdot X_\nu; \\ & S_2 = Y_1 \cdot X_1^2 + Y_2 \cdot X_2^2 + \ldots + Y_\nu \cdot X_\nu^2; \\ & \vdots \\ & S_{2 \cdot t} = Y_1 \cdot X_1^{2 \cdot t} + Y_2 \cdot X_2^{2 \cdot t} + \ldots + Y_\nu \cdot X_\nu^{2 \cdot t}. \end{aligned} \tag{3.43}$$

Згідно із визначенням поля Галуа над елементом α та циклотомічних класів, система рівнянь (3.43) зводиться до лінійної та має хоча б один розв'язок за умови $2 \cdot t < n$. Опишемо узагальнений метод пошуку розв'язку системи рівнянь (3.43) для довільного поля Галуа. Для цього використаємо штучний прийом, пов'язаний із введенням додаткових змінних, через які потім можна обчислити значення локаторів помилок.

Розглянемо відповідний поліном від змінної x:

$$\Lambda(x) = \Lambda_\nu x^\nu + \Lambda_{\nu-1} x^{\nu-1} + \ldots + \Lambda_1 x + 1, \tag{3.44}$$

який у теорії кодування називається поліномом локаторів помилок [52, 62, 63]. Головною властивістю поліному (3.44) є те, що його корені є зворотними елементами до локаторів помилок для значень $l = 1, 2, \ldots, \nu$, тобто:

$$x = X_l^{-1}. \tag{3.45}$$

Згідно із (3.45), перепишемо співвідношення (3.44) наступним чином:

$$\Lambda(x) = \left(1 - x \cdot X_1\right) \cdot \left(1 - x \cdot X_2\right) \cdot \ldots \cdot \left(1 - x \cdot X_\nu\right). \tag{3.46}$$

Тобто, за умови відомих коефіцієнтів $\Lambda_\nu, \cdots, \Lambda_1$ у співвідношенні (3.44), можна перетворити поліном локаторів помилок до форми (3.46) та знайти синдроми помилок $X_1, \ldots, X_\nu$. Таким чином, тепер головна задача декодування кодів БЧХ зводиться до обчислення за заданими коефіцієнтами синдрому помилки коефіцієнтів полінома локаторів помилок $\Lambda_\nu, \cdots, \Lambda_1$.

Для розв'язування цієї задачі необхідно зробити відповідні алгебраїчні перетворення. Спочатку помножимо обидві частини рівності (3.46) на добуток $Y_l \cdot X_l^{j+\nu}$. Тоді, враховуючи співвідношення (3.45), маємо [52, 62, 63]:

$$0 = Y_l \cdot X_l^{j+\nu} \cdot \left(1 + \Lambda_1 \cdot X_l^{-1} + \Lambda_2 \cdot X_l^{-2} + \ldots + \Lambda_{\nu-1} \cdot X_l^{\nu-1} + \Lambda_\nu \cdot X_l^{\nu}\right). \quad (3.47)$$

Перепишемо отримане співвідношення (3.47) в іншому вигляді. Після множення всіх складових поліному, записаного відносно локатора помилки X_l, на множник $X_l^{j+\nu}$, який у співвідношенні (3.47) винесений за дужки, отримуємо наступну тотожність:

$$0 = Y_l \cdot \left(X_l^{j+\nu} + \Lambda_1 \cdot X_l^{j+\nu-1} + \Lambda_2 \cdot X_l^{j+\nu-2} + \ldots + \Lambda_{\nu-1} \cdot X_l^{j-1} + \Lambda_\nu \cdot X_l^{j}\right). \quad (3.48)$$

Отримані співвідношення (3.48) є вірними для будь-яких значень *l* та *j*. Якщо просумувати ці тотожності за змінною *l* від 1 до ν, для кожного значення *j* маємо [52, 62, 63]:

$$\sum_{l=1}^{\nu} Y_l \cdot \left(X_l^{j+\nu} + \Lambda_1 \cdot X_l^{j+\nu-1} + \Lambda_2 \cdot X_l^{j+\nu-2} + \ldots + \Lambda_{\nu-1} \cdot X_l^{j-1} + \Lambda_\nu \cdot X_l^{j}\right) = 0. \quad (3.49)$$

Розкриваючи дужки у співвідношенні (3.49), перепишемо його наступним чином [63]:

$$\sum_{l=1}^{\nu} Y_l \cdot X_l^{j+\nu} + \Lambda_1 \cdot \sum_{l=1}^{\nu} Y_l \cdot X_l^{j+\nu-1} + \Lambda_2 \cdot \sum_{l=1}^{\nu} Y_l \cdot X_l^{j+\nu-2} + \ldots$$

$$\ldots + \Lambda_{\nu-1} \cdot \sum_{l=1}^{\nu} Y_l \cdot X_l^{j-1} + \Lambda_\nu \cdot \sum_{l=1}^{\nu} Y_l \cdot X_l^{j} = 0. \quad (3.50)$$

Оскільки у рівнянні (3.50) кожна сума є компонентою синдрому помилки, його можна переписати у вигляді:

$$S_{j+\nu} + \Lambda_1 \cdot S_{j+\nu-1} + \Lambda_2 \cdot S_{j+\nu-2} + \ldots + \Lambda_{\nu-1} \cdot S_{j-1} + \Lambda_\nu \cdot S_j = 0,$$

або в іншій формі [63]:

$$\Lambda_1 \cdot S_{j+\nu-1} + \Lambda_2 \cdot S_{j+\nu-2} + \ldots + \Lambda_{\nu-1} \cdot S_{j-1} + \Lambda_\nu \cdot S_j = -S_{j+\nu}, \quad (3.51)$$

$$j = 1, \ldots, \nu.$$

Отримана система лінійних рівнянь (3.51) зв'язує компоненти синдрому помилки S_l із коефіцієнтами полінома локатора помилки (3.44). Перепишемо

систему (3.51) у матричній формі [63]:

$$\begin{bmatrix} S_1 & S_2 & S_3 & \dots & S_{\nu-1} & S_\nu \\ S_2 & S_3 & S_4 & \dots & S_\nu & S_{\nu+1} \\ S_3 & S_4 & S_5 & \dots & S_{\nu+1} & S_{\nu+2} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ S_\nu & S_{\nu+1} & S_{\nu+2} & \dots & S_{2\cdot\nu-2} & S_{2\cdot\nu-1} \end{bmatrix} \cdot \begin{bmatrix} \Lambda_\nu \\ \Lambda_{\nu-1} \\ \Lambda_{\nu-2} \\ \dots \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} -S_{\nu+1} \\ -S_{\nu+2} \\ -S_{\nu+3} \\ \dots \\ -S_{2\cdot\nu} \end{bmatrix}. \qquad (3.52)$$

Якщо матриця синдромів помилок є невиродженою, систему лінійних рівнянь (3.52) можна розв'язати шляхом обернення цієї матриці. У цьому і полягає сутність метода ПГЦ. Тому тепер необхідно довести, що матриця синдромів помилок є невиродженою за умови кількості помилок ν, меншої за *t*. Розглянемо відповідну теорему матричного аналізу [63].

Теорема 3.4. Матриця Вандермонда, яка визначається як довільна матриця виду

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & \dots 1 \\ X_1 & X_2 & \dots X_\mu \\ X_1^2 & X_2^2 & \dots X_\mu^2 \\ \dots & \dots & \dots \dots \\ X_1^{\mu-1} & X_2^{\mu-1} & \dots X_\mu^{\mu-1} \end{bmatrix} \qquad (3.53)$$

має відмінний від нуля визначник тоді та лише тоді, коли виконується тотожність $\forall((i=1,\dots,\mu),(i=1,\dots,\mu),(i\neq j))(X_i \neq X_j)$.

Противне твердження є зрозумілим, оскільки у разі виконання тотожності $(X_i = X_j)$ у матриці Вандермонда (3.53) існує два однакових стовпця.

Олександр Теофіл Вандермонд
(1735 – 1796)

Для доведення прямого твердження теореми 3.3 скористаємося методом математичної індукції, який у загальному вигляді був розглянутий у підрозділі 7.1 другої частини посібника.

Зрозуміло, що твердження теореми 3.3 є вірним за умови $\mu = 1$. Тепер покажемо, що якщо твердження цієї теореми є вірним для матриці (3.53) із розмірністю $(\mu - 1) \times (\mu - 1)$, воно також є вірним для матриці розмірності $\mu \times \mu$.

Поперш за все запишемо визначник матриці (3.53) як транспоновану матрицю, у який замінемо змінну X_1 на незалежну змінну x. Тоді визначник матриці Вандермонда **A** можна розглядати як функцію від змінної x та записати його у наступному вигляді [63]:

$$\det(\mathbf{A}(x)) = \det\begin{bmatrix} 1 & x & x^2 & \dots & x^{\mu-1} \\ 1 & X_2 & X_2^2 & \dots & X_2^{\mu-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & X_\mu & X_\mu^2 & \dots & X_\mu^{\mu-1} \end{bmatrix}. \tag{3.54}$$

Із матричного співвідношення (3.54) зрозуміло, що визначник матриці **A** можна записати у вигляді поліному степені $\mu - 1$ наступним чином [63]:

$$\det(\mathbf{A}(x)) = d_{\mu-1}x^{\mu-1} + \ldots + d_1 x + d_0. \tag{3.55}$$

У співвідношенні (3.55) коефіцієнт $d_{\mu-1}$ не дорівнює нулю, у противному випадку матриця **A** є виродженою. Тепер припустимо, що для будь-якого елементу матриці (3.54) виконується умова:

$$x = X_i,\ i \neq 1. \tag{3.56}$$

Зрозуміло, що у разі виконання умови (3.56) стовпчики матриці (3.54) співпадають, а це означає, що її визначник дорівнює нулю. Тобто, умова (3.56) суперечить початковій умові поставленої задачі. У цьому разі отриманий поліном (3.55) можна розкласти на елементарні множники, виносячи постійну величину $d_{\mu-1}$ за дужки. Запис співвідношення (3.55) через лінійні множники має наступний вигляд [63]:

$$\det(\mathbf{A}(x)) = d_{\mu-1} \cdot \prod_{i=2}^{\mu}(x - X_i). \tag{3.57}$$

Враховуючи початкову умову $x = X_1$, співвідношення (3.57) можна переписати у вигляді [63]:

$$\det(\mathbf{A}(x)) = d_{\mu-1} \cdot \prod_{i=2}^{\mu}(X_1 - X_i). \tag{3.58}$$

Аналіз отриманого співвідношення (3.58) дозволяє зробити висновок, що визначник матриці **A** не дорівнює нулю. Цей висновок випливає з того, що, як було відмічено раніше, множник $d_{\mu-1}$ не дорівнює нулю внаслідок невиродженості матриці **A**, а всі коефіцієнти X_i є різними, оскільки в противному випадку стовпчики матриці (3.54) співпадають, тобто, вона також стає виродженою. Тепер теорему 3.3 можна вважати доведеною. ■

Із доведеної теореми 3.4 безпосередньо випливає головна теорема алгоритму ПГЦ для кодів БЧХ. Сформулюємо її наступним чином [52, 62, 63].

Теорема 3.5. Матриця локаторів помилок кодів БЧХ

$$\mathbf{M} = \begin{bmatrix} S_1 & S_2 & \dots & S_{\mu} \\ S_2 & S_3 & \dots & S_{\mu+1} \\ S_3 & S_4 & \dots & S_{\mu+2} \\ \dots & \dots & \dots & \dots \\ S_{\mu} & S_{\mu+1} & \dots & S_{2\cdot\mu-1} \end{bmatrix}, \tag{3.59}$$

задана у співвідношенні (3.52), є невиродженою, якщо розмірність матриці μ дорівнює кількості помилок у кодовій комбінації ν. Навпаки, матриця синдромів помилок (3.59) є виродженою, якщо виконується умова $\mu > \nu$.

Для доведення теореми 3.5 спочатку припустимо, що за умови $\mu > \nu$ $X_{\mu} = 0$ та задана матриця Вандермонда **A**, яка визначається співвідношенням (3.53). Зрозуміло, що елементи матриці Вандермонда A_{ij} дорівнюють

$$A_{ij} = X_j^{i-1}. \tag{3.60}$$

Визначимо також діагональну матрицю **B** з елементами

$$B_{ij} = Y_i X_i \delta_{ij}, \qquad \delta_{ij} = \begin{cases} 1, & \text{якщо } i = j; \\ 0, & \text{якщо } i \neq j, \end{cases}$$

тобто

$$\mathbf{B} = \begin{bmatrix} Y_1 X_1 & 0 & \dots & 0 \\ 0 & Y_1 X_1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & Y_\mu X_\mu \end{bmatrix}. \tag{3.61}$$

Розрахуємо тепер елементи матриці

$$\mathbf{F} = \mathbf{A}\mathbf{B}\mathbf{A}^{\mathrm{T}}. \tag{3.62}$$

Згідно із співвідношеннями (3.53), (3.60) – (3.62), маємо [63]:

$$F_{ij} = \sum_{l=1}^{\mu} X_l^{i-1} \sum_{k=1}^{\mu} X_l Y_l \delta_{lk} X_k^{i-1} = \sum_{l=1}^{\mu} X_l^{i-1} Y_l X_l X_l^{i-1} = \sum_{l=1}^{\mu} Y_l X_l^{i+j-1}. \tag{3.63}$$

Із співвідношень (3.59), (3.63), можна зробити висновок, що матриці **F** та **M** співпадають [63], тобто

$$\mathbf{F} = \mathbf{M}. \tag{3.64}$$

Наслідком отриманої тотожності (3.64) та співвідношення (3.62) є те, що визначник матриці **M** можна розрахувати через визначники матриць **A** та **B**, заданих співвідношеннями (3.53), (3.60) – (3.62) [63]:

$$\det(\mathbf{M}) = \det(\mathbf{A}) \cdot \det(\mathbf{B}) \cdot \det(\mathbf{A}) = \det^2(\mathbf{A}) \cdot \det(\mathbf{B}). \tag{3.65}$$

Також із співвідношення (3.61), записаного для матриці **B**, випливає, що [63]:

$$\det(\mathbf{B}) = 0, \text{ якщо } \mu > \nu;$$
$$\det(\mathbf{B}) \neq 0, \text{ якщо } \mu \leq \nu,$$

звідки, враховуючи (3.65):

$$\det(\mathbf{M}) = 0, \text{ якщо } \mu > \nu;$$
$$\det(\mathbf{M}) \neq 0, \text{ якщо } \mu \leq \nu. \tag{3.66}$$

Тобто, отримане остаточне співвідношення (3.66), відповідно з яким теорему 3.5 можна вважати доведеною. ■

На основі доведеної теореми 3.5 будується алгоритм ПГЦ, який застосовується для в на практиці для декодування кодів БЧХ [52, 62, 63]. Цей алгоритм формується наступним чином. Спочатку необхідно знайти вірне значення кількості помилок у зчитаній кодовій послідовності ν. Як перше можливе значення вводиться величина $\nu = t$, де t – максимальна можлива

кількість помилок. Якщо за цієї умови визначник матриці синдромів помилок **M**, яка визначається співвідношенням (3.59), дорівнює 0, необхідно зменшити величину ν на 1 та повторити операцію обчислення визначника матриці **M**. Коли визначник стає відмінним від нуля, розв'язується матрична система рівнянь (3.52). Розв'язок цієї системи у матричній формі записується наступним чином [52, 62, 63]:

$$\begin{bmatrix} \Lambda_{\nu} \\ \Lambda_{\nu-1} \\ \Lambda_{\nu-2} \\ \dots \\ \Lambda_1 \end{bmatrix} \cdot \mathbf{M^{-1}} = \begin{bmatrix} -S_{\nu+1} \\ -S_{\nu+2} \\ -S_{\nu+3} \\ \dots \\ -S_{2\cdot\nu} \end{bmatrix}. \tag{3.67}$$

Далі, за умови відомих коефіцієнтів поліному локаторів помилок Λ_i, розв'язується поліноміальне рівняння (3.44) та, через значення векторів синдромів помилок X_i шукаються вірні символи кодової комбінації Y_i. Матричне рівняння, за допомогою якого обчислюються вірні значення символів кодової комбінації, записується через матрицю Вандермонда (3.53) наступним чином:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ \dots \\ Y_{\nu} \end{bmatrix} \cdot \begin{bmatrix} X_1 & X_2 & X_3 & \dots & X_{\nu} \\ X_1^2 & X_2^2 & X_3^2 & \dots & X_{\nu}^2 \\ X_1^3 & X_2^3 & X_3^3 & \dots & X_{\nu}^3 \\ \dots & \dots & \dots & \dots & \dots \\ X_1^{\nu} & X_2^{\nu} & X_3^{\nu} & \dots & X_{\nu}^{\nu} \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ \dots \\ S_{\nu} \end{bmatrix}. \tag{3.68}$$

Слід відзначити, що для двійкових кодів БЧХ задача пошуку помилки значно спрощується, оскільки локатор помилки точно описує позиції помилок, а вірному значенню символу Y_i відповідає інверсне значення помилкового символу X_i. Тобто, для двійкового коду БЧХ необхідно розв'язати лише матричну систему рівнянь (3.52), а розв'язок системи рівнянь (3.68) є тривіальним:

$$Y_i = \overline{X_i}\,,\ \mathrm{i} = 1, \dots, \nu. \tag{3.69}$$

Тепер в описаному алгоритмі декодування кодів БЧХ найбільш складною задачею є пошук коренів X_i многочлену $\Lambda(x)$ (3.44), оскільки ця задача поки не формалізована нами з алгоритмічної точки зору. Зазвичай під

час реалізації для алгоритму ПГЦ для пошуку коренів многочлену $\Lambda(x)$ застосовують просту, але досить ефективну процедуру Ченя.

Сутність процедури Ченя полягає у тому, що для многочлену $\Lambda(x)$ для кожного значення j послідовно обчислюються значення $\Lambda\left(\alpha^j\right)$ та як корені полінома обираються ті з них, для яких виконується умова $\Lambda\left(\alpha^j\right)=0$.

Узагальнена блок-схема алгоритму ПГЦ наведена на рис. 3.7 [63].

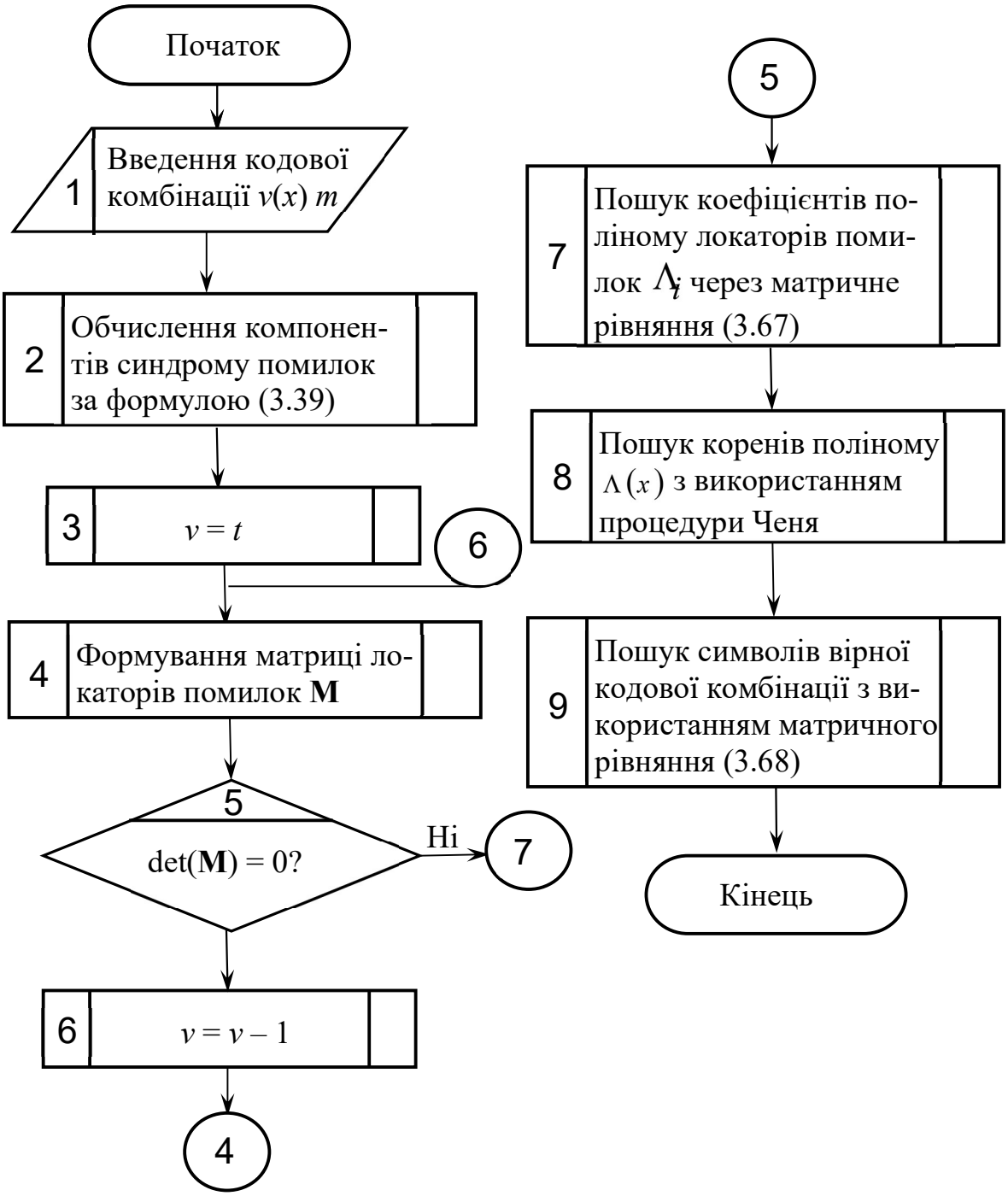


Рис. 3.7 Блок-схема алгоритму ПГЦ для декодування кодів БЧХ

Щодо процедури Ченя, існує декілька можливих алгоритмів пошуку коренів поліному $\Lambda(x)$, проте найбільш ефективним з них з точки зору мінімальної кількості обчислювальних операцій вважається використання схеми Горнера з прямою підстановкою змінної. Цей метод базується на теорії поліномів, яка розглядалася у третьому розділі другої частини посібника [48]. Розкладання поліному $\Lambda(x)$ через допоміжну змінну β можна записати наступним чином [63]:

$$\Lambda(\beta)=((((\Lambda_{\nu}\cdot\beta+\Lambda_{\nu-1})\cdot\beta+\Lambda_{\nu-2})\cdot\beta+\Lambda_{\nu-3})\cdot\beta+\ldots+\Lambda_0). \qquad (3.70)$$

Із формули (3.70) видно, що для обчислення значення поліному $\Lambda(\beta)$ за схемою Горнера необхідно виконати ν операцій множення та ν операцій додавання. Така поліноміальна операція може бути легко реалізована на обчислювальних електронних пристроях, схеми яких були розглянуті у підрозділі 3.7 другої частини посібника [48]. Більш досконало особливості роботи електронних кодувальних схем, основаних на поліноміальних операціях, розглянуті у навчальних посібниках [52, 78]. Взагалі робота обчислювальних пристроїв кодувальних схем базується на теорії скінченних автоматів, яка розглядалася у підрозділі 7.3 другої частини посібника [80].

Крім розглянутої схеми Горнера, можливою є також пряма підстановка значень β^i до поліному (3.44), якщо наперед відома та занесена до пам'яті електронного обчислювального пристрою таблиця цих значень. Кількість операцій множення та додавання у цьому разі не змінюється, проте необхідно виконати $\nu - 2$ додаткові операції звернення до комірок пам'яті. Виконання алгебраїчної операції піднесення до степені є досить складним для практичної реалізації в електронній апаратурі і в обчислювальних алгоритмах вона зазвичай не розглядається як елементарна [52, 78].

Розглянемо приклад пошуку подвійної помилки у коді БЧХ (15, 5) з використанням алгоритму ПГЦ [63].

Приклад 3.13. З використанням алгоритму ПГЦ знайти подвійну помилку у коді БЧХ (15, 5), вектор якої заданий поліномом $v(x) = x^7 + x^2$.

Спосіб побудови п’ятнадцятирозрядного коду БЧХ, який виправляє потрійні помилки, був розглянутий у прикладі 3.5. Відповідний твірний поліном задається співвідношенням (3.27) та має вигляд:

$$g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1.$$

Для проведення подальших обчислень за алгоритмом ПГЦ скористаємося формулою формування синдромів помилок (3.43). Синдроми формуються наступним чином. Поліном $v(x)$ записуються через степені змінної елемента поля Галуа α. Оскільки для коду БЧХ, який розглядається, $t = 3$, необхідно записати у полі Галуа $GF(16)$ шість синдромів помилок, тобто, від першої степені поліному $v(x)$ до шостої.

З одного боку зрозуміло, що оскільки кількість помилок ν є меншою за значення t, поліном $v(x)$ відповідає значенню помилки $e(x)$, тобто, $v(x) = e(x)$. Проте на початковому етапі декодер не може зробити такого висновку. Необхідно створити рівняння для локаторів помилок, проаналізувати їх через матрицю Вандермонда, знайти її визначник у поліноміальній формі, потім з використанням процедури Ченя знайти значення коефіцієнтів поліному локаторів помилок. Саме ці значення надають декодеру інформацію про положення помилкових розрядів, а у разі використання багаторозрядних кодів – також про величини помилок, що надає можливість їх виправляти [63]. У цьому і полягає сутність алгоритму ПГЦ, описаного у цьому підрозділі, блок-схема якого наведена на рис. 3.7.

Тобто, почнемо з формування рівнянь синдромів помилок для поліному прийнятої кодової комбінації $v(x)$ згідно із співвідношенням (3.43). Для обчислення суми степенів слід використовувати алгебраїчні операції над елементами поля Галуа $GF(16)$ через поле $GF(2)$. Тоді відповідну систему рівнянь можна записати наступним чином:

$$S_1 = \alpha^7 + \alpha^2 = \alpha^{12};\ S_2 = \alpha^{14} + \alpha^4 = \alpha^9;\ S_3 = \alpha^{21} + \alpha^6 = 0; \qquad (3.71)$$

$$S_4 = \alpha^{28} + \alpha^8 = \alpha^3;\ S_5 = \alpha^{35} + \alpha^{10} = \alpha^0; S_6 = \alpha^{42} + \alpha^{24} = 0.$$

Тепер, згідно із блок-схемою алгоритму, наведеною на рис. 3.7, з використанням співвідношенням (3.71) необхідно сформувати матрицю

синдромів помилок **M** як матрицю Вандермонда. За умови ν = 3 матриця **M** записується наступним чином:

$$\mathbf{M}_3 = \begin{bmatrix} S_1 & S_2 & S_3 \\ S_2 & S_3 & S_4 \\ S_3 & S_4 & S_5 \end{bmatrix} = \begin{bmatrix} \alpha^{12} & \alpha^9 & 0 \\ \alpha^9 & 0 & \alpha^3 \\ 0 & \alpha^3 & 1 \end{bmatrix}. \qquad (3.72)$$

Оскільки головна діагональ матриці $\mathbf{M}_3$ дорівнює нулю, її визначних також дорівнює нулю. Тобто, ця матриця є виродженою, що свідчить про те, що загальна кількість помилок у коді є меншою за 3. Тому, на основі рівнянь синдромів помилок (3.71), запишемо матрицю $\mathbf{M}_2$ для двох помилок у кодовій комбінації $v(x)$. Відповідна матриця має вигляд:

$$\mathbf{M}_2 = \begin{bmatrix} S_1 & S_2 \\ S_2 & S_3 \end{bmatrix} = \begin{bmatrix} \alpha^{12} & \alpha^9 \\ \alpha^9 & 0 \end{bmatrix}. \qquad (3.73)$$

Оскільки визначник матриці $\mathbf{M}_2$ $\det(\mathbf{M}_2) \neq 0$, робимо висновок, що у кодовій комбінації $v(x)$ сталося дві помилки. Тепер, згідно із алгоритмом, наведеним на рис. 3.7, необхідно знайти матрицю $\mathbf{M}_2^{-1}$, зворотну до матриці $\mathbf{M}_2$, задану співвідношенням (3.73). Використовуючи алгебраїчні операції над полем Галуа $GF(16)$, можна записати:

$$\mathbf{M}_2^{-1} = \begin{bmatrix} 0 & \alpha^{15-9} \\ \alpha^{15-9} & \alpha^{15-6} \end{bmatrix} = \begin{bmatrix} 0 & \alpha^6 \\ \alpha^6 & \alpha^9 \end{bmatrix}. \qquad (3.74)$$

Тепер, за умови відомої зворотної матриці локаторів помилок (3.74), з використанням співвідношення (3.67) можна записати матричне рівняння для коефіцієнтів поліному локаторів помилок:

$$\begin{bmatrix} \Lambda_2 \\ \Lambda_1 \end{bmatrix} = \begin{bmatrix} 0 & \alpha^6 \\ \alpha^6 & \alpha^9 \end{bmatrix} \cdot \begin{bmatrix} S_3 \\ S_4 \end{bmatrix} = \begin{bmatrix} 0 & \alpha^6 \\ \alpha^6 & \alpha^9 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ \alpha^3 \end{bmatrix} = \begin{bmatrix} \alpha^9 \\ \alpha^6 \cdot \alpha^6 \end{bmatrix} = \begin{bmatrix} \alpha^9 \\ \alpha^{12} \end{bmatrix}. \qquad (3.75)$$

Враховуючи отриманий результат (3.75), поліном локаторів помилок записується у вигляді:

$$\Lambda(x) = \alpha^9 \cdot x^2 + \alpha^{12} \cdot x + 1. \qquad (3.76)$$

Із отриманого співвідношення (3.76), застосовуючи схему Горнера для процедури Ченя, задану співвідношенням (3.70), можна записати наступне розкладання для поліному синдромів помилок:

$$\Lambda(x) = \alpha^9 \cdot \left(x - \alpha^8\right) \cdot \left(x - \alpha^{13}\right). \qquad (3.77)$$

Тобто, многочлен локаторів помилок має два корені: α^8 та α^{13}. Тоді самі локатори помилок є величинами, оберненими до цих значень. Використовуючи алгебраїчні операції над елементами поля Галуа *GF*(16), остаточно маємо [48, 55]:

$$\left(\alpha^8\right)^{-1} = \alpha^{15-8} = \alpha^7; \qquad \left(\alpha^{13}\right)^{-1} = \alpha^{15-13} = \alpha^2. \tag{3.78}$$

Враховуючи (3.78), вектор помилок можна записати у вигляді:

$$e(x) = v(x) = x^7 + x^2.$$

Як було відмічено у підрозділі 3.3.4.1, алгоритм ПГЦ є наочним та зрозумілим, але не ефективним для реалізації в електронних апаратних та програмованих кодувальних пристроях. Це пов'язано з тим, що алгебраїчні операції над групами та поліноми над ними досить важко формалізувати, вони більше пристосовані для проведення ручних розрахунків. Саме тому алгоритм ПГЦ історично був розроблений першим та сприяв подальшому бурхливому розвитку теорії кодування та практики використання групових кодів [52, 62, 63]. Сьогодні в електронній кодувальній апаратурі частіше використовується більш ефективний ітераційний алгоритм Берлекемпа – Мессі, який буде розглянутий у наступному підрозділі. Структурні схеми обчислювальних електронних пристроїв для декодування кодів БЧХ розглядатимуться у підрозділі 3.3.5. особливості використання алгоритму ПГЦ для декодування послідовностей кодів Ріда – Соломона розглядатимуться у підрозділі 3.4.7.

Тобто, алгоритм ПГЦ варто використовувати лише для пошуку у кодах БЧХ невеликої кількості помилок, зазвичай меншою за 3 [62, 63]. За такої умови для пошуку розв'язку матричних систем рівнянь (3.52) та (3.53) застосовуються відомі методди матричного аналізу, наприклад, метод пошуку оберненої матриці або метод виключення змінних Гауса – Зейделя. Приклади такого використання алгоритму ПГЦ для систематичних кодів Ріда – Соломона, які можна розглядати як один із різновидів кодів БЧХ, будуть наведені у підрозділах 3.4.7 та 3.4.9.

3.3.4.3 Алгоритм Берлекемпа – Мессі та теорема Форні

Зрозуміло, що головним недоліком розглянутого у підрозділі 3.3.4.2 алгоритму ПГЦ є необхідність обернення двох матриць із розмірністю $t \times t$. Процедура обернення матриць є складною з обчислювальної точки зору, і тому в обчислювальних алгоритмах її намагаються уникати. У цьому підрозділі розглядатимуться ітераційні обчислювальні алгоритми, які дозволяють

здійснювати декодування кодів БЧХ без реалізації процедури обернення матриць. Обернення першої матриці, (3.59), яке є необхідним для обчислення коефіцієнтів поілному локаторів помилок, можна уникнути, застосовуючи алгоритм Берлекемпа – Мессі. А обернення другої матриці, заданої рівнянням (3.68), яке є необхідним для обчислення значень помилки, вдається уникнути з використанням алгоритму Форні. Слід відзначити, що оскільки у двійкових кодах значення помилки завжди є однозначним та визначається співвідношенням (3.69), друга процедура обернення матриці (3.68) для таких кодів взагалі є зайвою. Саме для двійкових кодів БЧХ алгоритм ПГЦ і був початково запропонований У. Пітерсоном. Проте алгоритм Берлекемпа – Мессі дозволяє цілком уникнути процедури обернення матриць, яка з обчислювальної точки зори є досить ресурсоємною [52, 62, 63]. Теорема Форні та алгоритм Берлекемпа – Мессі, а також головні особливості декодування двійкових кодів БЧХ з використанням алгоритму Берлекемпа – Мессі, розглядатимуться у цьому підрозділі.

Для описання алгоритму Форні розглянемо та проаналізуємо поліном локаторів помилки (3.44) із коренями X_l^{-1}, тобто:

$$\Lambda(x)=\prod_{i=1}^{\nu}(1-x\cdot X_l). \tag{3.79}$$

Для поліному локаторів помилок, записаного у формі (3.79), поліном синдромів помилок можна записати наступним чином [52, 62, 63]:

$$S(x)=\sum_{j=1}^{2\cdot t} S_j\cdot x^j=\sum_{j=1}^{2\cdot t}\sum_{i=1}^{\nu} Y_i\cdot X_i^j\cdot x^j\ . \tag{3.80}$$

Для поліному значень помилок можна записати наступний вираз [52, 62, 63]:

$$\Omega(x)=\mathrm{mod}\Big((S(x)\cdot\Lambda(x)),\left(x^{2\cdot t}\right)\Big). \tag{3.81}$$

Поліноми локаторів та значень помилок, записані у вигляді співвідношень (3.80) та (3.81), будуть використані у подальших міркуваннях. Розглянемо важливу теорему теорії кодування, яка встановлює зв’язок між поліномами локаторів та значень помилок [52, 62, 63].

Теорема 3.6. Поліном значень помилок (3.81), з урахуванням виразів (3.79), (3.80), можна записати у вигляді:

$$\Omega(x)=\sum_{i=1}^{\nu} Y_i\cdot X_i\cdot\prod_{l\neq i}(1-x\cdot X_l). \tag{3.82}$$

Дійсно, враховуючи співвідношення (3.79) – (3.81), можна записати:

$$\Omega(x)=\left[\sum_{j=1}^{2\cdot t}\sum_{i=1}^{\nu} Y_i\cdot X_i^j\cdot x^j\right]\cdot\left[\prod_{i=1}^{\lambda}(1-X_i\cdot x)\right]\left(\mathrm{mod}\left(x^{2\cdot t}\right)\right)=$$

$$=\sum_{i=1}^{\nu} Y_i\cdot X_i^j\cdot\left[(1-X_i\cdot x)\cdot\sum_{j=1}^{2\cdot t}(X_i\cdot x)^{j-1}\right]\cdot\prod_{i=1}^{\lambda}(1-X_l\cdot x)\left(\mathrm{mod}\left(x^{2\cdot t}\right)\right). \tag{3.83}$$

У записаному співвідношенні (3.83) вираз у дужках можна розглядати як розкладання у степеневий ряд виразу $\left(1-X_i^{2\cdot t}\cdot x^{2\cdot t}\right)$. Відповідно, маємо:

$$\Omega(x)=\sum_{i=1}^{\nu} Y_i\cdot X_i^j\cdot\left[(1-X_i\cdot x)\cdot\sum_{j=1}^{2\cdot t}(X_i\cdot x)^{j-1}\right]\cdot\prod_{i=1}^{\lambda}(1-X_l\cdot x)\left(\mathrm{mod}\left(x^{2\cdot t}\right)\right)=$$

$$=\sum_{i=1}^{\nu} Y_i\cdot X_i\cdot\prod_{l\neq i}(1-x\cdot X_l).$$

Тобто, розглянуту теорему 3.6 можна вважати доведеною. ■

Із доведеної теореми 3.6 безпосередньо випливає наступна теорема, яка є теоретичним підґрунтям для реалізації алгоритму Форні [52, 62, 63].

Теорема 3.7. (Теорема Форні). Значення помилок у коді БЧХ можна розрахувати із співвідношення:

$$Y_l=\frac{X_l^{-1}\cdot\Omega\left(X_l^{-1}\right)}{\prod_{l\neq i}\left(1-X_j\cdot X_l^{-1}\right)}=\frac{\Omega\left(X_l^{-1}\right)}{\Lambda'\left(X_l^{-1}\right)}. \tag{3.84}$$

Доведення теореми 3.7, з урахуванням результату, отриманого в теоремі 3.6, є досить простим. Дійсно, після підстановки до виразу (3.82) значення $x=X_l^{-1}$, отримуємо:

$$\Omega\left(X_l^{-1}\right)=Y_l\cdot X_l\cdot\prod_{l\neq i}\left(1-X_j\cdot X_l^{-1}\right),$$

звідки

$$Y_l = \frac{X_l^{-1} \cdot \Omega\left(X_l^{-1}\right)}{\prod\limits_{l \neq i}\left(1 - X_j \cdot X_l^{-1}\right)}.$$

Тобто, перша частина рівності (3.84) доведена.

Для доведення другої частини теореми 3.7 необхідно взяти похідну від виразу (3.79) за змінною x та підставити до отриманого виразу значення $x = X_l^{-1}$. Відповідно маємо:

$$\Lambda'(x) = -\sum_{i=1}^{\nu} X_i \cdot \prod_{j \neq i}\left(1 - x \cdot X_j\right),$$

таким чином:

$$\Lambda'\left(X_l^{-1}\right) = -\sum_{i=1}^{\nu} X_i \cdot \prod_{j \neq i}\left(1 - x \cdot X_l^{-1}\right).$$

Тобто, розглянуту теорему 3.7 можна вважати доведеною.

Тепер розглянемо ітераційний алгоритм Берлекемпа – Мессі, який дозволяє без здійснення операції обернення матриць знаходити помилки у кодах БЧХ. Запишемо цей алгоритм спочатку у вигляді тезових формулювань, а потім розглянемо блок-схему цього алгоритму [52, 62, 63]. Слід відзначити, що узагальнений алгоритм декодування Берлекемпа – Мессі для багатопозиційних кодів БЧХ є досить складним, тому в кінці цього підрозділу буде розглянутий спрощений варіант цього алгоритму для двійкових кодів. У вигляді тезових формулювань алгоритм Берлекемпа – Мессі можна записати наступним чином.

1. Введення початкових значень поліному помилок $\Lambda(x) = 1$ та допоміжного поліному $B(x) = 1$, а також значень кількості контрольних розрядів $r = 0$ та довжини кодувального регістру $L = 0$.

2. Збільшення кількості контрольних розрядів r на 1, тобто $r = r + 1$.

3. Обчислення значення помилки у відповідній компоненті синдрому:

$$\Delta_r = S_r + \sum_{j=1}^{L} \Lambda_j \cdot S_{r-j} = \sum_{j=0}^{L} \Lambda_j \cdot S_{r-j}. \qquad (3.85)$$

4. Якщо $\Delta_r = 0$, перехід до пункту алгоритму 13, а у противному випадку – до пункту 5.

5. Обчислення нового поліному зв'язків, для якого $\Delta_r = 0$, через співвідношення:

$$T(x) = \Lambda(x) - \Delta_r \cdot x \cdot B(x). \tag{3.86}$$

6. Якщо $2 \cdot L \leq r - 1$, перехід до пункту 7, у противному випадку – перехід до пункту 12.

7. Збереження попереднього значення допоміжного поліному $B(x)$ після нормалізації:

$$B(x) = \Delta_r^{-1} \Lambda(x). \tag{3.87}$$

8. Заміна поліному $\Lambda(x)$ на $T(x)$, тобто:

$$\Lambda(x) = T(x). \tag{3.88}$$

9. Обчислення нового значення довжини кодувального регістру:

$$L = r - L. \tag{3.89}$$

10. Перехід до пункту 13.

11. Обчислення нового значення поліному $\Lambda(x)$ за формулою (3.88).

12. Обчислення нового значення поліному $B(x)$ за формулою:

$$B(x) = x \cdot B(x). \tag{3.90}$$

13. За умови $r = 2 \cdot t$, перехід до пункту алгоритму 2, у противному випадку – до пункту 14.

14. Обчислення порядку поліному синдрому помилок $\Lambda(x)$:

$$c = \deg(\Lambda(x)), \tag{3.91}$$

15. За умови $c = L$ вважати що синдром помилки задається поліномом $\Lambda(x)$ та обчислити значення помилок за формулою (3.84). У противному випадку вважати, що кодова послідовність є хибною через кількість помилок ν є більшою за значення t.

15. Кінець проведення розрахунків.

Блок-схема алгоритму Берлекемпа – Мессі наведена на рис. 3.8.

Розглянемо приклад роботи алгоритму Берлекемпа – Мессі для коду БЧХ (15, 5) [63].

Приклад 3.14. З використанням алгоритму Берлекемпа – Мессі знайти помилку у коді БЧХ (15, 5) для прийнятої кодової комбінації $v(x) = e(x) = x^7 + x^5 + x^2$.

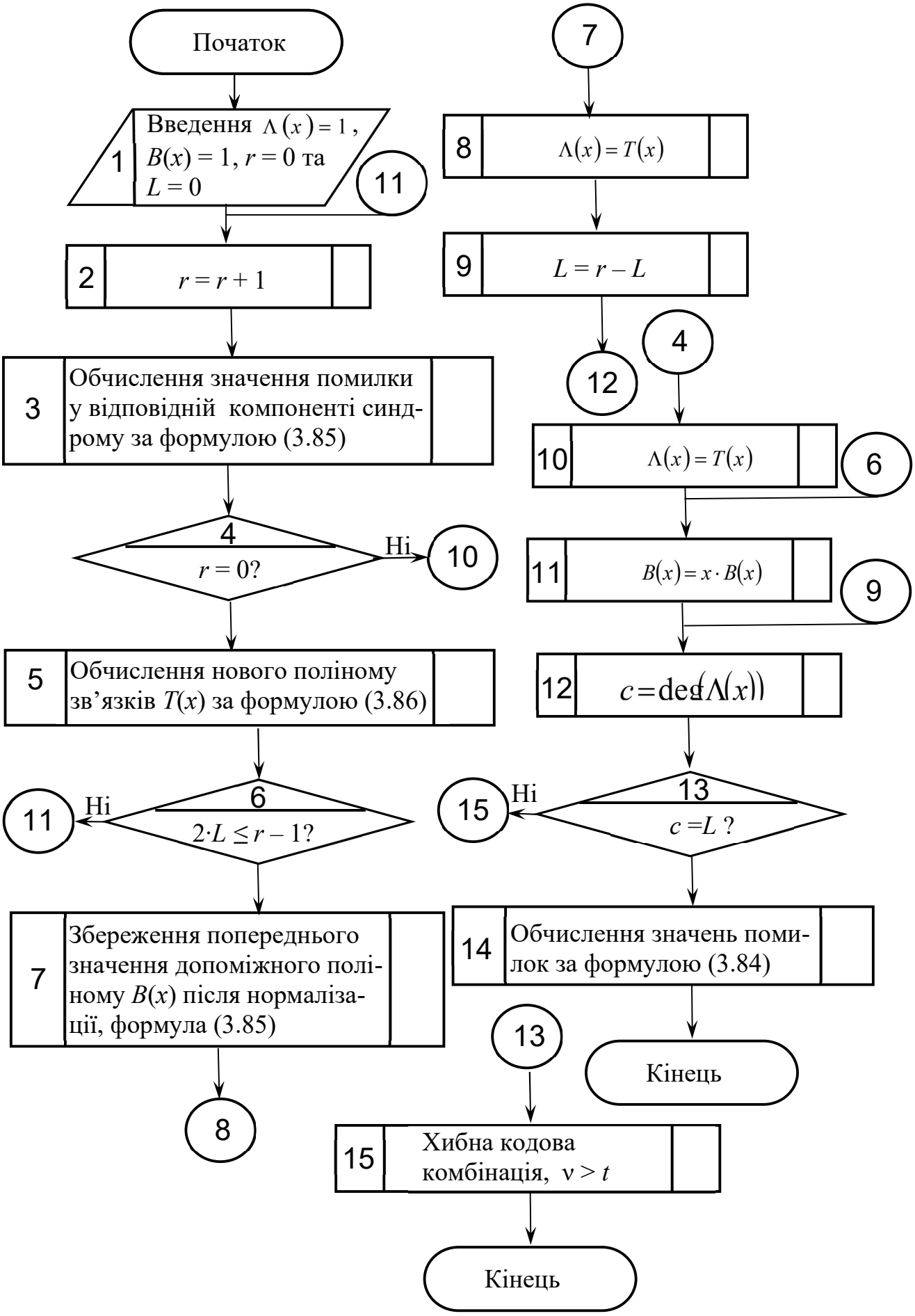


Рис. 3.8 Блок-схема алгоритму Берлекемпа – Мессі

Як і у прикладі 3.13, запишемо для вектора прийнятої кодової комбінації $v(x) = x^7 + x^5 + x^2$ породжувальний поліном та контрольні суми:

$g(x) = x^{10} + x^8 + x^5 + x^2 + x^4 + x^2 + x + 1;$

$S_1 = \alpha^7 + \alpha^5 + \alpha^2 = \alpha^{14}; \quad S_2 = \alpha^{14} + \alpha^{10} + \alpha^4 = \alpha^{13}; \; S_3 = \alpha^{21} + \alpha^{15} + \alpha^6 = 1;$

$S_4 = \alpha^{28} + \alpha^{20} + \alpha^8 = \alpha^{11}; \; S_5 = \alpha^{35} + \alpha^{25} + \alpha^{10} = \alpha^5; \; S_6 = \alpha^{42} + \alpha^{30} + \alpha^{12} = 1.$

Результати ітераційних обчислень за алгоритмом Берлекемпа – Мессі наведені у таблиці 3.3.

Таблиця 3.3 – Результати розрахунків за алгоритмом Берлекемпа – Мессі для прикладу 3.14

r	Δ_r	$T(x)$	$B(x)$	$\Lambda(x)$	L
0	–	–	1	1	0
0	α^{14}	$1 + \alpha^{14} \cdot x$	α	$1 + \alpha^{14} \cdot x$	1
2	0	$1 + \alpha^{14} \cdot x$	$\alpha \cdot x$	$1 + \alpha^{14} \cdot x$	1
3	α^{11}	$1 + \alpha^{14} \cdot x + \alpha^{12} \cdot x^2$	$\alpha^4 + \alpha^3 \cdot x$	$1 + \alpha^{14} \cdot x + \alpha^{12} \cdot x^2$	2
4	0	$1 + \alpha^{14} \cdot x + \alpha^{12} \cdot x^2$	$\alpha^4 + \alpha^3 \cdot x^2$	$1 + \alpha^{14} \cdot x + \alpha^{12} \cdot x^2$	2
5	α^{11}	$1 + \alpha^{14} \cdot x + \alpha^{12} \cdot x^2 + \alpha^{14} \cdot x^3$	$\alpha^4 + \alpha^3 \cdot x^2 + \alpha \cdot x^2$	$1 + \alpha^{14} \cdot x + \alpha^{12} \cdot x^2 + \alpha^{14} \cdot x^3$	3
6	0	$1 + \alpha^{14} \cdot x + \alpha^{12} \cdot x^2 + \alpha^{14} \cdot x^3$	$\alpha^4 \cdot x + \alpha^3 \cdot x^2 + \alpha \cdot x^3$	$1 + \alpha^{14} \cdot x + \alpha^{12} \cdot x^2 + \alpha^{14} \cdot x^3$	3

Тобто, поліному синдрому помилок $\Lambda(x)$ записується у вигляді:

$$\Lambda(x) = 1 + \alpha^{14} \cdot x + \alpha^{12} \cdot x^2 + \alpha^{14} \cdot x^3 = (1 + \alpha^7 \cdot x) \cdot (1 + \alpha^5 \cdot x) \cdot (1 + \alpha^2 \cdot x),$$

що відповідає синдрому помилок $v(x)$.

Як було відмічено вище, алгоритм Берлекемпа – Мессі, блок-схема якого наведена на рис. 3.8, є вірною для будь-якого скінченного поля Галуа $GF(m)$ і є досить узагальненою, цим і обумовлена відносна складність наведеного алгоритму.

Для двійкових кодів БЧХ алгоритм декодування Берлекемпа – Мессі та ітераційні співвідношення (3.85) – (3.91) значно спрощується. Насамперед це спрощення обумовлене тим, що необхідно фіксувати лише позиції помилок, а

їх значення дорівнює 1. Тому значення помилки завжди однозначно визначається співвідношенням (3.69).

Інше спрощення полягає в особливості формування контрольних сум для синдромів помилок, на яку зверталась увага у підрозділі 3.1. Річ у тому, що на парних ітераціях значення $\Delta_r = 0$, що пов'язано з тим, що для поля Галуа *GF*(2) рівняння для синдромів S_1 та S_2 пов'язані між собою. Відповідні аналітичні перетворення, із яких слідує це твердження, також були наведені у підрозділі 3.1. Тому для двійкових кодів синдроми помилок можна записати наступним чином [52, 62, 63]:

$$\Delta_1 = \mathrm{S}_1; \qquad \Delta_3 = \mathrm{S}_1 + \mathrm{S}_2; \qquad \Lambda^{(1)}(x) = S_1 \cdot x + 1;$$

$$\Lambda^{(3)}(x) = \left(\frac{S_3}{S_1} + S_2 \right) \cdot x^2 + S_1 \cdot x + 1. \qquad (3.92)$$

Для поліному синдрому помилок порядку *r*, згідно із співвідношеннями (3.85) – (3.91), за умови використання двійкових кодів БЧХ, можна записати:

$$\Lambda^{(r)}(x) = \Lambda^{(r-2)}(x) - \Delta_r \cdot x^2 B^{(r-2)}(x);$$

$$B^{(r)}(x) = \delta_r \cdot \Delta_r^{-1} \cdot \Lambda^{(r-2)}(x) + (1 - \delta_r) \cdot x^2 \cdot B^{(r-2)}(x). \qquad (3.93)$$

Слід також відзначити, що на практиці в електронних пристроях зазвичай використовуються систематичні коди БЧХ, спосіб створення яких є аналогічним способу формування систематичних циклічних кодів, які розглядалися у підрозділі 2.5.5. Декодери систематичних циклічних кодів, принцип функціонування яких оснований на алгоритмі Берлекемпа – Мессі, будуть розглядатися у підрозділі 3.3.6. Структурні схеми електронних обчислювальних декодувальних пристроїв, у яких реалізований алгоритм Берлекемпа – Мессі, розглядатимуться також у підрозділі 3.3.5.

Алгоритм Берлекемпа – Мессі також ефективно використовується для декодування послідовностей групових кодів Ріда – Соломона. Спосіб такого використання цього алгоритму, а також відповідні приклади, розглядатимуться у підрозділі 3.4.6.

3.3.4.4 Алгоритм Евкліда

Алгоритм Евкліда загалом оснований на пошуку найменшого спільного дільника двох поліномів. Зазвичай у теорії поліномів цей алгоритм вважається не дуже ефективним [52, 62], проте його ефективність, як і ефективність будь-якого іншого алгоритму, завжди суттєво залежить від конкретних умов практичного застосування [52, 62, 63].

Розглянемо відповідні теореми теорії поліномів [52, 62, 63].

Теорема 3.8. (Алгоритм Евкліда для поліномів). Нехай задані два поліноми $s(x)$ та $t(x)$, для яких $\deg(s(x)) \geq \deg(t(x))$. Будемо також вважати, що на початковій ітерації $s^{(0)}(x) = s(x)$, $t^{(0)}(x) = t(x)$. Задамо тепер початкову матрицю **A** у вигляді $\mathbf{A}^{(0)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Тоді розв'язком рекурентних рівнянь:

$$Q^{(r)}(x) = \left[\frac{s^{(r)}(x)}{t^{(r)}(x)}\right]; \qquad \mathbf{A}^{(r+1)}(x) = \begin{bmatrix} 0 & 1 \\ 1 & -Q^{(r)}(x) \end{bmatrix} \cdot \mathbf{A}^{(r)}(x);$$

$$\begin{bmatrix} s^{(r+1)}(x) \\ t^{(r+1)}(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -Q^{(r)}(x) \end{bmatrix} \cdot \begin{bmatrix} s^{(r)}(x) \\ t^{(r)}(x) \end{bmatrix} = \mathbf{A}^{(r+1)}(x) \cdot \begin{bmatrix} s^{(r)}(x) \\ t^{(r)}(x) \end{bmatrix} \tag{3.94}$$

є поліном

$$s^{(R)}(x) = \gamma \cdot \text{НСД}[s(x), t(x)]. \tag{3.95}$$

Крім цього, завжди існує така величина γ, для якої виконується тотожність:

$$\gamma \cdot \text{НСД}[s(x), t(x)] = A_{11}^{(R)}(x) \cdot s(x) + A_{12}^{(R)}(x) \cdot t(x), \tag{3.96}$$

де R має таке значення, що $t^{(R)}(x) = 0$.

Доведемо теорему 3.8. По-перш за все необхідно відмітити, що вона цілком аналогічна теоремі Евкліда для теорії чисел, яка розглядалася у підрозділі 1.1.3 другої частини посібника.

Спочатку слід врахувати умову $\deg(t^{(r+1)}(x)) < \deg(t^{(r)}(x))$. За цією умовою завжди існує таке значення R, для якого $t^{(R)}(x) = 0$, і ітераційний процес, заданий співвідношеннями (3.94), на цьому етапі обривається. Враховуючи це, можна записати наступний вираз [63]:

$$\begin{bmatrix} s^{(R)}(x) \\ 0 \end{bmatrix} = \left\{ \prod_{r=R-1}^{0} \begin{bmatrix} 0 & 1 \\ 1 & -Q^{(r)}(x) \end{bmatrix} \right\} \cdot \begin{bmatrix} s(x) \\ t(x) \end{bmatrix}. \tag{3.97}$$

Із отриманого співвідношення (3.97) безпосередньо випливає, що спільний дільник поліномів $s(x)$ та $t(x)$ також є дільником поліному $s^{(R)}(x)$.

Враховуючи, що [63]:

$$\begin{bmatrix} 0 & 1 \\ 1 & -Q^{(r)}(x) \end{bmatrix}^{-1} = \begin{bmatrix} Q^{(r)}(x) & 1 \\ 1 & 0 \end{bmatrix}, \tag{3.98}$$

перепишемо співвідношення (3.97), з урахуванням (3.98), у наступному вигляді:

$$\begin{bmatrix} s^{(R)}(x) \\ 0 \end{bmatrix} = \left\{ \prod_{r=R-1}^{0} \begin{bmatrix} Q^{(r)}(x) & 1 \\ 1 & 0 \end{bmatrix} \right\} \cdot \begin{bmatrix} s^{(R)}(x) \\ t(x) \end{bmatrix}. \tag{3.99}$$

Аналізуючи отримане співвідношення (3.99) можна зробити висновок, що поліном $s^{(R)}(x)$ є найбільшим спільним дільником поліномів $s(x)$ та $t(x)$. Таким чином, виконується співвідношення (3.95) [63].

З іншого боку, з урахуванням співвідношень (3.94) та (3.97), можна записати, що [63]:

$$\begin{bmatrix} s^{(R)}(x) \\ 0 \end{bmatrix} = A^{(R)}(x) \begin{bmatrix} s(x) \\ t(x) \end{bmatrix},$$

тобто:

$$s^{(R)}(x) = A_{11}^{(R)}(x) \cdot s(x) + A_{12}^{(R)}(x) \cdot t(x).$$

Таким чином, співвідношення (3.96) також виконується, і теорему 3.7 можна вважати доведеною. ■

Із доведення теореми 3.8 зрозуміло, яке значення мають елементи матриці **A**, A_{11} та A_{12}, заданої ітераційними співвідношеннями (3.94). Аналогічно можна знайти значення двох інших елементів матриці **A**. Розглянемо відповідну теорему теорії поліномів [63].

Теорема 3.9. (Наслідок теореми 3.8). Поліноми A_{11} та A_{12} для матриці **A**, заданої ітераційними співвідношеннями (3.94), задовольняють наступним умовам:

$$s(x) = (-1)^R A_{22}^{(R)}(x) \cdot \gamma \cdot \text{НСД}[s(x), t(x)];$$

$$t(x)=(-1)^R\cdot A_{21}^{(R)}(x)\cdot\gamma\cdot\text{НСД}[s(x),t(x)]. \qquad (3.100)$$

Дійсно, враховуючи вираз (3.98) для зворотної матриці $\mathbf{A}^{-1}$, можна записати:

$$\begin{bmatrix} s(x) \\ t(x) \end{bmatrix} = (-1)^R \cdot \begin{bmatrix} A_{22}^{(R)}(x) & A_{21}^{(R)}(x) \\ -A_{21}^{(R)}(x) & A_{11}^{(R)}(x) \end{bmatrix} \cdot \begin{bmatrix} s^{(R)}(x) \\ 0 \end{bmatrix}, \qquad (3.101)$$

звідки безпосередньо випливають тотожності (3.100) [63]. Тобто, теорему 3.9 також можна вважати доведеною. ■

Розглянемо один із можливих способів практичного використання алгоритму Евкліда для декодування кодів БЧХ [63].

Повернемося до рівняння (3.81), яке встановлює зв'язок між поліномами синдромів помилок, локаторів помилок та їхніми значеннями. Нагадаємо, що використовувалось нами під час розгляду алгоритму ПГЦ. Також врахуємо умови, які визначають степені поліномів локаторів та значень помилок, а саме:

$$\deg(\Omega(x))=t, \quad \deg(\Lambda(x))=t-1. \qquad (3.102)$$

Тепер спробуємо використати алгоритм Евкліда, заданий співвідношеннями (3.94), для розв'язування порівняння (3.81) відносно поліномів $\Omega(x)$ та $\Lambda(x)$. Перепишемо ітераційне рівняння (3.94) у вигляді [52, 63]:

$$\begin{bmatrix} s^{(r)}(x) \\ t^{(r)}(x) \end{bmatrix} = \begin{bmatrix} A_{11}^{(r)}(x) & A_{12}^{(r)}(x) \\ A_{21}^{(r)}(x) & A_{22}^{(r)}(x) \end{bmatrix} \cdot \begin{bmatrix} s(x) \\ 0 \end{bmatrix},$$

звідки [63]:

$$t^{(r)}(x)=t(x)\cdot A_{22}^{(r)}(x)(\text{mod}(s(x))). \qquad (3.103)$$

На першій ітерації будемо вважати, що

$$t(x)=S(x),\ s(x)=x^{2\cdot t}. \qquad (3.104)$$

Тоді розв'язок алгебраїчного рівняння (3.103) з урахуванням початкових умов (3.104) зводиться до того, що необхідно знайти значення *r*, для якого виконуються умови [52, 63]:

$$\deg\left(A_{22}^{(r)}(x)\right)\le t, \quad \deg\left(t^{(r)}(x)\right)\le t-1. \qquad (3.105)$$

Із (3.105) випливають наступні ітераційні співвідношення для поліномів $t^{(r)}$:

$$\deg\left(t^{(r-1)}(x)\right)\geq t\,,\ \ \deg\left(t^{(r)}(x)\right)\leq t-1\,. \tag{3.106}$$

Враховуючи (3.103), зрозуміло, що на початковій ітерації

$$\deg\left(t^{(0)}(x)\right)=2\cdot t\,. \tag{3.107}$$

Зрозуміло також, що степінь поліному $t^{(r)}(x)$ спадає із збільшенням значення r, а із цього безпосередньо випливає, що, з урахуванням (3.107), системі нерівностей (3.106) задовольняє лише єдине значення r', для якого:

$$\deg\left(t^{(r')}(x)\right)\leq t-1\,. \tag{3.108}$$

З іншого боку, степінь поліному $A_{22}^{(r)}$ зростає із збільшенням номера ітерації r, тому залишається лише показати, що

$$\deg\left(A_{22}^{(r')}(x)\right)\leq t\,. \tag{3.109}$$

Це буде означати, що рівняння (3.103) має єдиний розв'язок r'.

Доведемо справедливість тотожності (3.109) методом обернення матриці **A** [52, 63]. Спочатку скористаємося співвідношенням (3.97), з якого випливає, що [63]:

$$A^{(r')}(x)=\prod_{r=r'-1}^{0}\begin{bmatrix}0 & 1\\ 1 & -Q^{(r)}(x)\end{bmatrix}. \tag{3.110}$$

Із записаного співвідношення (3.110) зрозуміло, що поліном $A_{22}^{(r)}=-Q^{(r)}(x)$ має найбільшу степінь серед всіх коефіцієнтів матриці **A**. Тобто, вірною є наступна тотожність:

$$\deg\left(A_{22}^{(r)}(x)\right)\geq\deg\left(A_{12}^{(r)}(x)\right). \tag{3.111}$$

Також слід мати на увазі, що степінь поліному $s^{(r)}(x)$ завжди перевищує степінь поліному $t^{(r)}(x)$, тобто:

$$\deg\left(s^{(r)}(x)\right)>\deg\left(t^{(r)}(x)\right). \tag{3.112}$$

Враховуючи співвідношення (3.111), (3.112), а також матричну рівність (3.101), яку можна переписати у вигляді:

$$\begin{bmatrix} s(x) \\ t(x) \end{bmatrix} = (-1)^{r'} \cdot \begin{bmatrix} A_{22}^{(r)}(x) & A_{21}^{(r)}(x) \\ -A_{21}^{(r)}(x) & A_{11}^{(r)}(x) \end{bmatrix} \cdot \begin{bmatrix} s^{(r)}(x) \\ t^{(r)}(x) \end{bmatrix}, \tag{3.113}$$

можна знайти нижню граничну величину для степені поліному $A_{22}^{(r)}(x)$.

Дійсно, із матричного рівняння (3.113) безпосередньо випливає, що [63]:

$$\deg(s(x)) = \deg\left(A_{22}^{(r')}(x)\right) + \deg\left(s^{(r')}(x)\right). \tag{3.114}$$

Тепер, враховуючи те, що співвідношення алгоритму Евкліда (3.94) є ітераційними, можна записати наступну тотожність:

$$\deg\left(s^{(r')}(x)\right) = \deg\left(t^{(r'-1)}(x)\right). \tag{3.115}$$

Тоді, з урахуванням отриманих співвідношень (3.114), (3.115), можна остаточно записати наступну оцінку для нижньої граничної величини степені поліному $A_{22}^{(r)}(x)$:

$$\deg\left(A_{22}^{(r')}(x)\right) = \deg(s(x)) - \deg\left(t^{(r'-1)}(x)\right) \le 2 \cdot t - t = t. \tag{3.116}$$

Згідно із співвідношеннями (3.94) – (3.116), поліноми $\Lambda(x)$ та $\Omega(x)$ можна знайти із наступних співвідношень [52, 62, 63]:

$$\Delta = A_{22}^{(r')}(0);\ \Omega(x) = \Delta^{-1} \cdot t^{(r')}(x);\ \Lambda(x) = \Delta^{-1} \cdot A_{22}^{(r')}(x). \tag{3.117}$$

Як було відмічено раніше, алгоритм Евкліда для поліномів є повністю аналогічним алгоритму Евкліда для чисел, розглянутому у підрозділі 1.1.3 другої частини посібника. Матричні співвідношення (3.94) можна переписати у вигляді наступної системи рівнянь [52]:

$$\begin{cases} s_0(x) = Q_1(x) \cdot t_0(x) + R_1(x); \\ t_0(x) = Q_2(x) \cdot R_1(x) + R_2(x); \\ R_1(x) = Q_2(x) \cdot R_1(x) + R_2(x); \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots; \\ R_{r-3}(x) = Q_{r-1}(x) \cdot R_{r-2}(x) + R_{r-1}(x); \\ R_{r-2}(x) = Q_r(x) \cdot R_{r-1}(x). \end{cases} \tag{3.118}$$

Тобто, сутність цього алгоритму полягає у послідовному діленні поліномів та взятті остач. Розглянемо відповідні приклади.

Приклад 3.15. З використанням алгоритму Евкліда знайти найбільший спільний дільник поліномів $s(x) = x^3 + 3\cdot x^2 + 3\cdot x + 2$ та $t(x) = x^3 + 2\cdot x^2 + 2\cdot x + 1$.

Ітераційний процес ділення поліномів та взяття остач показаний на рис. 3.9.

Рис. 3.9. Наочна ілюстрація ітераційного процесу ділення поліномів за алгоритмом Евкліда для прикладу 3.15

Згідно із рис. 3.9 найбільшим спільним дільником поліномів $s(x)$ та $t(x)$ є остання остача, яка не дорівнює 0, тобто, поліном $x^2 + x + 1$. Дійсно, як видно з рис. 3.10, поліном $x^2 + x + 1$ є дільником обох поліномів, визначених в умові задачі.

$$\begin{array}{r|l} x^3+3\cdot x^2+3\cdot x+2 & x^2+x+1 \\ \hline -\,x^3+x^2+x & x+2 \\ \hline -\,2\cdot x^2+2\cdot x+2 & \\ \hline -\,2\cdot x^2+2\cdot x+2 & \\ \hline 0 & \end{array} \qquad \begin{array}{r|l} x^3+2\cdot x^2+2\cdot x+1 & x^2+x+1 \\ \hline -\,x^3+x^2+x & x+1 \\ \hline x^2+x+1 & \\ \hline -\,x^2+x+1 & \\ \hline 0 & \end{array}$$

Рис. 3.10 Ділення поліномів $x^3 + 3\cdot x^2 + 3\cdot x + 2$ та $x^3 + 2\cdot x^2 + 2\cdot x + 1$ на спільний дільник $x^2 + x + 1$

Аналогічно проводиться пошук найбільшого спільного дільника для двійкових поліномів, проте операцію віднімання слід замінити на сумування за модулем 2. Розглянемо відповідний приклад.

Приклад 3.16. З використанням алгоритму Евкліда знайти найбільший спільний дільник поліномів $s(x) = x^4 + x^3 + x + 1$ та $t(x) = x^3 + 1$.

Результат ділення $\left[\dfrac{s(x)}{t(x)}\right]$ дає результат, показаний на рис. 3.11.

$$\begin{array}{r|l} x^4+x^3+x+1 & x^3+1 \\ \hline \oplus\; x^4+x^3 & x+1 \\ \hline x+1 & \end{array}$$

Рис. 3.11 Результат ділення двійкових поліномів за алгоритмом Евкліда для прикладу 3.16

З рис. 3.11 зрозуміло, що найбільшим спільним дільником поліномів $x^4 + x^3 + x + 1$ та $x^3 + 1$ є поліном $x + 1$. Дійсно, як видно з рис. 3.12, остача від ділення поліномів $x^4 + x^3 + x + 1$ та $x^3 + 1$ на поліном $x + 1$ дорівнює нулю.

$$\begin{array}{r|l} x^4 + x^3 + x + 1 & x + 1 \\ \oplus\ x^4 + x^3 & \hline x^3 + 1 \\ \hline x + 1 & \\ \oplus\ x + 1 & \\ \hline 0 & \end{array} \qquad \begin{array}{r|l} x^3 + 1 & x + 1 \\ \oplus\ x^3 + x^2 & \hline x^2 + x + 1 \\ \hline x^2 + 1 & \\ \oplus\ x^2 + x & \\ \hline x + 1 & \\ \oplus\ x + 1 & \\ \hline 0 & \end{array}$$

Рис. 3.12 Ділення двійкових поліномів $x^4 + x^3 + x + 1$ та $x^3 + 1$ на спільний дільник $x + 1$

Слід відзначити, що, як і в теорії чисел, алгоритм Евкліда в теорії поліномів тісно пов'язаний із методами формування ланцюгових дробів. Це пов'язано з тим, що частку двох остач $\frac{r_{n-1}}{r_n}$, які були отримані на сусідніх ітераціях, можна записати у вигляді ланцюгового дробу [52]:

$$\frac{r_{n-1}}{r_n} = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ddots \\ \ldots + \cfrac{1}{a_n}}}}}. \tag{3.119}$$

Більш досконало зв'язок теорії поліномів із алгоритмом Евкліда розглянутий у монографіях Е. Берлекемпа [52] та У. Пітерсона [62].

Блок-схема алгоритму Евкліда для декодування кодів БЧХ, заданого співвідношеннями (3.94), показана на рис. 3.13. Після того, як знайдені

поліноми локаторів помилок $\Lambda(x)$ та значень помилок $\Omega(x)$, можна завершити ітераційну процедуру пошуку помилок з використанням одного з методів, які були запропоновані для алгоритмів ПГЦ або Берлекемпа – Мессі, наприклад, процедурою Форні. Неефективність використання алгоритму Евкліда в кодувальній електронній апаратурі пов'язана із необхідністю реалізації на апаратному рівні ресурсоємної операції ділення двійкових поліномів.

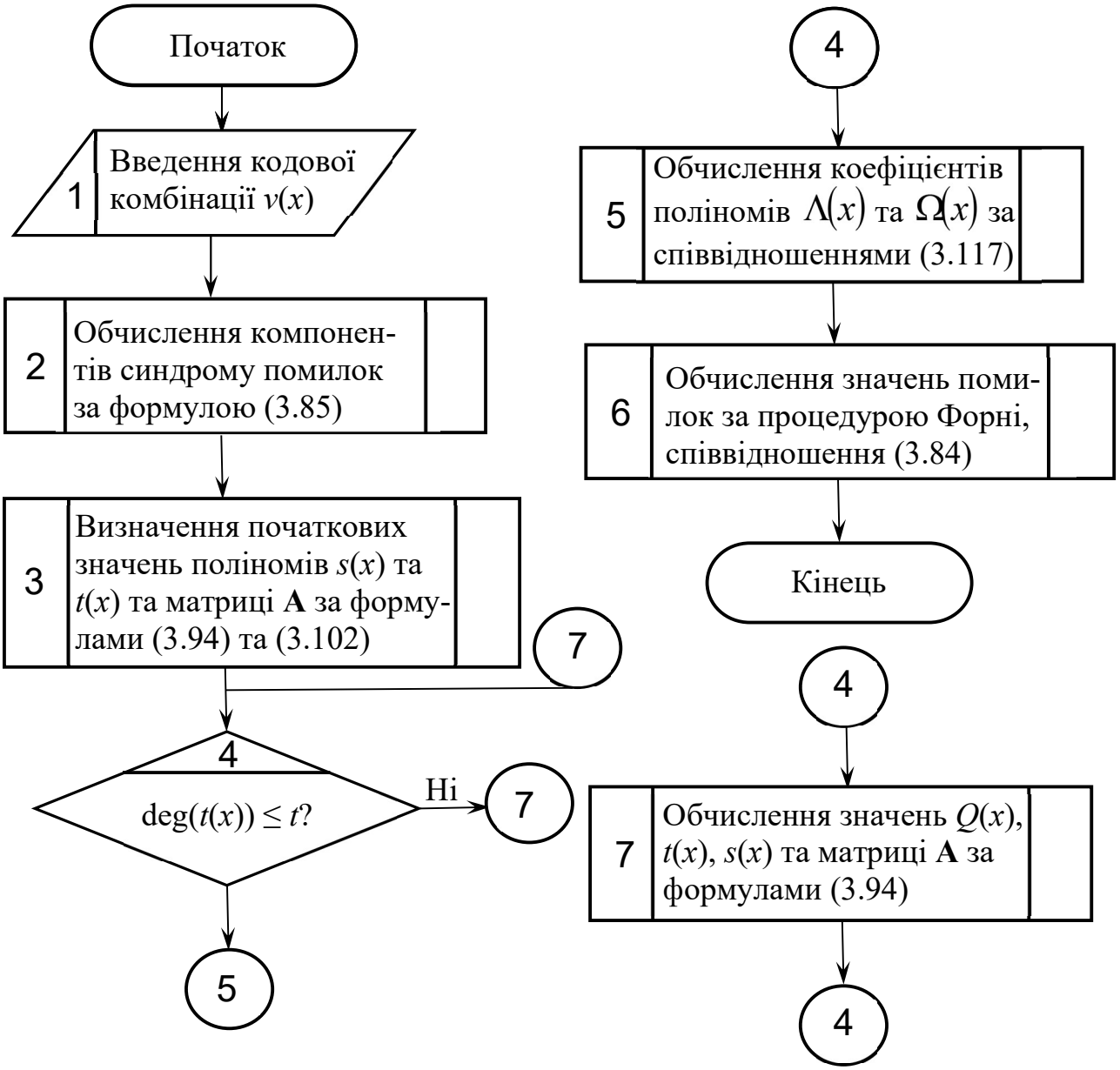


Рис. 3.13 Модифікація алгоритму Евкліда для декодування кодів БЧХ

3.3.5 Особливості апаратної реалізації обчислювальних схем для декодерів кодів Боуза – Чоудхурі – Хоквінгема

Загалом декодувальні електронні схеми для кодів БЧХ основані на алгоритмах їхнього декодування, описаних у попередньому підрозділі, та, як і декодери циклічних кодів, будуються на основі таких елементів цифрової електроніки, як регістри зсуву, регістри пам'яті та схеми сумування за модулем два. Більшість із цих схем є загальновідомими та описані у навчальній літературі [52, 62, 63]. Розглянемо декілька узагальнених варіантів апаратної реалізації декодерів кодів БЧХ.

У загальному випадку всі декодери кодів БЧХ базуються на чисельному розв'язуванні матричної системи рівнянь (3.52). Проте алгоритм ПГЦ, розглянутий у підрозділі 3.3.4.2 та оснований на обернені матриці **M**, в електронних кодувальних пристроях майже не використовується. Це пов'язано з тим, що чисельна процедура обернення матриць є дуже ресурсоємною, і за умови збільшення розмірності матриці ν кількість операцій множення та додавання у полі Галуа, необхідних для обчислення оберненої матриці, зростає пропорційно ν^3 [63]. Особливості комп'ютерної реалізації алгебраїчних та поліноміальних операцій у полях Галуа із аналізом кодів відповідних програм будуть окремо розглянуті у підрозділі 3.4.4.

Сутність розглянутого у підрозділі 3.3.4.3 алгоритму Берлекемпа – Мессі полягає у тому, що матрична система рівнянь (3.52) розглядається з точки зору можливості побудови кодувальної схеми на регістрах із лінійними зворотними зв'язками. Такий підхід оснований на аналізі визначеної матричної системи рівнянь та на виявлені можливостей її спрощення. Припустимо, що вектор значень коефіцієнтів поліному локатора помилок Λ є заздалегідь відомим. Скористаємось тим, що матриця **M** будується як матриця Вандермонда та елементи S_i, які використані для її формування, відрізняються від елементів S_j, які формують вектор S, розташований у правій частині співвідношення (3.52). Тут важливим є те, що коефіцієнти S_i та S_j формуються послідовно та є незалежними. Тому систему рівнянь (3.52) можна переписати у вигляді [63]:

$$S_j = -\sum_{i=1}^{\nu} \Lambda_i \cdot S_{j-i}\,,\ j = \nu+1,\ldots,2\cdot\nu. \qquad (3.120)$$

У теорії кодування доведено, що для фіксованих значень Λ_i система рівнянь (3.120) описує авторегресійний фільтр, загальна теорія якого та принцип побудови розглядалися у підрозділі 6.5.4 другої частини посібника [49]. Аналогічні приклади обчислювальних електронних пристроїв для забезпечення роботи з двійковими поліномами були наведені у підрозділі 3.7 другої частини посібника. Структурна обчислювальна схема авторегресійного фільтру, побудованого на основі регістрів пам'яті та логічних елементів сумування за модулем два, наведена на рис. 3.14 [52, 62, 63]. Слід відзначити, що ця схема є узагальненою, і якщо деякі з коефіцієнтів поліному локаторів помилок для конкретного коду БЧХ дорівнюють нулю, відповідні зв'язки в ній пропускаються. Наприклад, для двійкових кодів БЧХ, згідно із співвідношенням (3.93), слід враховувати лише коефіцієнти із парними індексами.

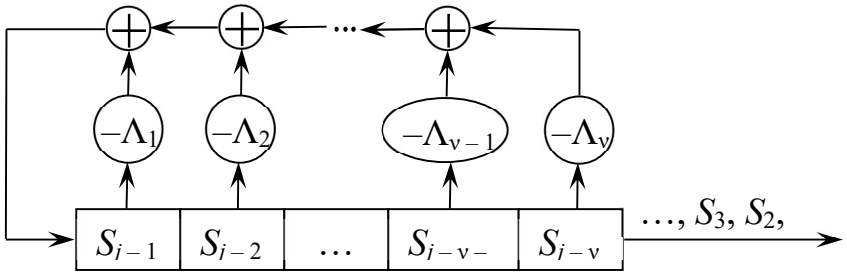


Рис. 3.14 Узагальнена обчислювальна схема авторегресійного фільтру, який виконує перетворення цифрового сигналу згідно із співвідношенням (3.120)

Для практичної реалізації декодерів кодів БЧХ, згідно із структурною схемою, наведеною на рис. 3.14, важливо знати значення коефіцієнтів Λ_i та загальну довжину регістру пам'яті L, яка у загальному випадку повинна бути більшою, ніж степінь поліному локаторів помилок $\Lambda(x)$, тобто [52, 63]:

$$L \geq \deg(\Lambda(x)). \qquad (3.121)$$

З урахуванням співвідношення (3.121), задача синтезу декодера кодів БЧХ зводиться до мінімізації довжини регістру пам'яті L. Для цього на кожній ітерації з номером r обчислюється розбіжність Δ_r згідно із співвідношенням:

$$S_j = \sum_{j=0}^{n-1} \Lambda_j^{(r-1)} \cdot S_{r-j} \,. \qquad (3.122)$$

Ідея роботи декодера за алгоритмом Берлекемпа – Мессі полягає у тому, що за умови $\Delta_r = 0$ можна вважати, що

$$(L_r, \Lambda^{(r)}(x)) = (L_{r-1}, \Lambda^{(r-1)}(x)) \qquad (3.123)$$

та закінчити поточну ітерацію r. У противному випадку необхідно змінити вагові множники у ланцюгу зворотного зв'язку наступним чином:

$$\Lambda^{(r)}(x) = \Lambda^{(r-1)}(x) + A \cdot x^l \cdot \Lambda^{(m-1)}(x), \qquad (3.124)$$

де A – елемент поля Галуа, l – ціле число, $\Lambda^{(m-1)}(x)$ – коефіцієнти поліному $\Lambda(x)$, які були розташовані у регістрі пам'яті на попередній ітерації.

Тепер, з використанням нового поліному $\Lambda^{(r)}(x)$, заданого співвідношенням (3.124), необхідно обчислити нове значення розбіжності Δ'_r:

$$\Delta'_r = \sum_{j=0}^{n-1} \Lambda_j^{(r)} \cdot S_{r-j} = \sum_{j=0}^{n-1} \Lambda_j^{(r-1)} \cdot S_{r-j} + A \cdot \sum_{j=0}^{n-1} \Lambda_j^{(m-1)} \cdot S_{r-j-1} \,. \qquad (3.125)$$

За таких умов визначаються коефіцієнти m, l та A. Будемо вважати, що:

$$m < r,\ \Delta_m \neq 0,\ l = r - m,\ A = -\Delta_m^{-1} \cdot \Delta_r \,. \qquad (3.126)$$

Враховуючи (3.125) та (3.126), можна записати:

$$\Delta'_r = \Delta_r - \frac{\Delta_r}{\Delta_m} \cdot \Delta_m = 0 \,. \qquad (3.127)$$

Із отриманого співвідношення (3.127) випливає, що кількість ітерацій r є достатньою для декодування будь-якої послідовності кодів БЧХ заданої коректувальної здатності. У теорії кодування доказана теорема про те, що для декодування послідовності із довжиною $2t$ достатнім є регістр із довжиною t [52, 63]. Проте, згідно із співвідношенням (3.126), для використання регістру пам'яті такої довжини необхідно мати додатковий цифровий пристрій, який обчислює елемент, зворотний до елементу A у заданому полі Галуа GF(2^m).

Одна із можливих конструкцій обчислювальних схем декодера, який працює за алгоритмом Берлекемпа – Мессі, наведена на рис. 3.15 [52, 62, 63]. На цій схемі електронний компонент, призначений для обчислювання значення зворотного елементу, позначений символом –1.

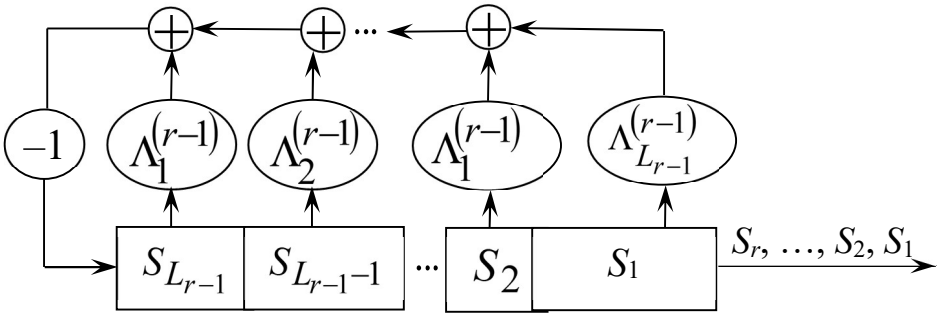


Рис. 3.15 Узагальнена обчислювальна схема декодера Берлекемпа – Мессі

Можливий робочий варіант узагальненої структурної схеми декодера Берлекемпа – Мессі, в якому для пошуку значень помилки використаний метод Форні, наведений на рис. 3.16 [63].

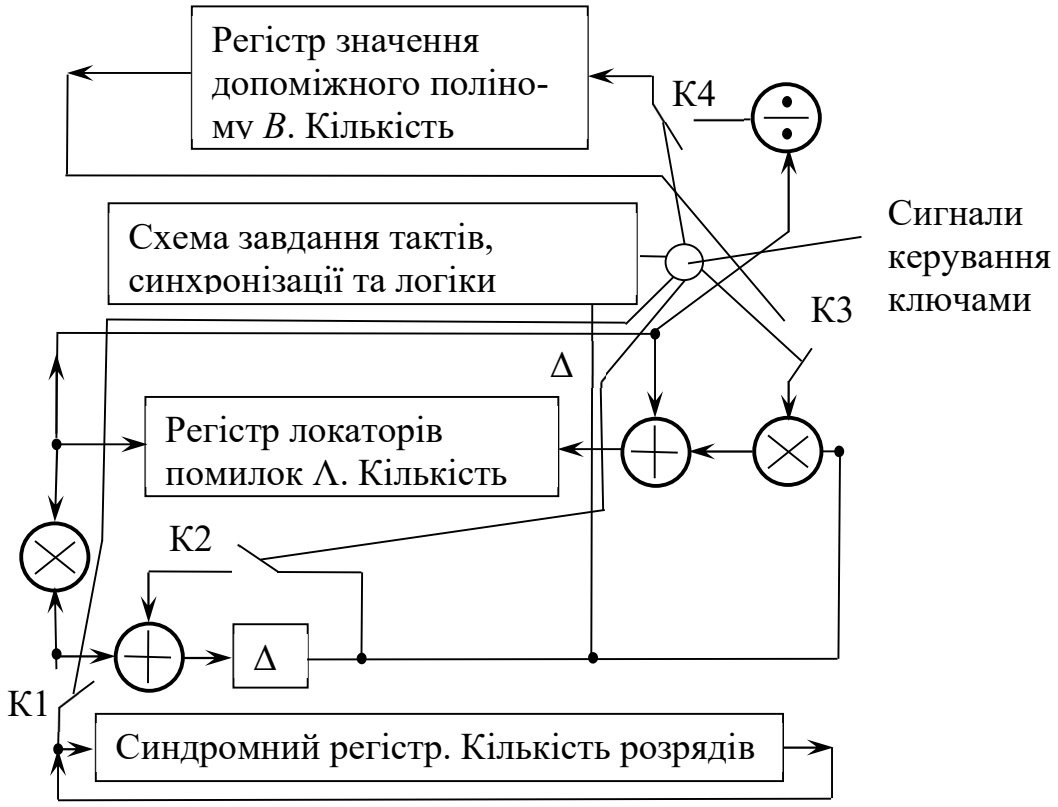


Рис. 3.16 Структурна схема декодера Берлекемпа – Мессі із системою синхронізації та блоком виправлення помилок

Розглянемо головні особливості роботи цієї схеми. Вона містить три регістри, призначені для зберігання коефіцієнтів поліномів $S(x)$, $\Lambda(x)$ та $B(x)$. Довжина кожного регістру повинна бути не менше тієї величини, яка є

необхідною для збереження всіх коефіцієнтів відповідного полінома. Якщо у кодових послідовностях використовуються поліноми менших порядків, відповідні старші розряди регістрів заповнюються нулями. Слід також відмітити, що регістри для поліномів $S(x)$ та $B(x)$ мають на один розряд більше.

В процесі роботи схеми на кожній ітерації здійснюється зсув всіх коефіцієнтів на одну позицію праворуч. Саме за рахунок цього забезпечується множення коефіцієнтів поліному $B(x)$ на змінну x та переіндексація компонентів S_j таким чином, щоб забезпечити розрахунок значень розбіжності за формулою (3.127). За рахунок такої індексації забезпечується правильне множення коефіцієнтів поліномів $S(x)$ та $\Lambda(x)$. Логіка роботи декодера та визначення часу зсуву для значень коефіцієнтів у регістрах організуються за допомогою відповідної логічної схеми, яка формує імпульси синхронізації та керує ключами К1 – К4. Наприклад, у регістрі значень поліному $\Lambda(x)$ зсув праворуч здійснюється двічі за один такт, а саме, під час обчислення розбіжності Δr та під час введення нового значення Λ.

Слід відзначити, що організація роботи схеми декодера Берлекемпа – Мессі, наведеної на рис. 3.16, цілком відповідає блок-схемі алгоритму, наведеній на рис. 3.8. Особливості функціональної логіки роботи цієї схеми можуть бути описані з використанням теорії скінченних автоматів, описаної у підрозділі 7.3 другої частини посібника [50], а також з використанням сучасних комп'ютерних засобів подійного моделювання процесів в електронних системах [49, 75 – 77].

Структурні схеми декодерів кодів БЧХ, наведені у цьому підрозділі, є досить узагальненими. Проте перевага таких схем полягає у тому, що вони є універсальненими і можуть бути використані для розробки цифрових електронних пристроїв, призначених для декодування як двійкових, так і багатопозиційних кодів. Реальні електронні схеми декодерів двійкових кодів БЧХ із різною коректувальною здатністю, які найчастіше використовуються в сучасній електронній апаратурі, будуть наведені у наступному підрозділі.

Проте, наведене у цьому підрозділі узагальнене описання принципів побудови декодерів БЧХ, складає теоретичне підгрунтя для аналізу особливостей роботи реальних декодувальних електронних схем та подальшого їхнього проектування [52, 53, 55 – 57, 62, 63].

3.3.6 Цифрові електронні пристрої для формування та декодування систематичних кодів Боуза – Чоудхурі – Хоквінгема із різною коректувальною здатністю

3.3.6.1 Формування та декодування систематичних кодів Боуза – Чоудхурі – Хоквінгема, які виправляють подвійні помилки

У підрозділі 3.3.3 розглядалися способи формування несистематичних кодів БЧХ, проте, як відмічалось у підрозділі 2.5.5, головний недолік несистематичних циклічних кодів під час їхньої реалізації в електронній кодувальній апаратурі полягає у тому, що в отриманій кодовій комбінації неможливо окремо виділити інформаційні розряди. Саме з цієї причини в електронних системах та у системах зв'язку частіше використовуються систематичні коди БЧХ, які дозволяють вирішити цю проблему [33, 56, 57]. Алгоритм побудови систематичних кодів БЧХ є досить простим, оскільки у теорії кодування коди БЧХ розглядаються як циклічні. Тому, за умови відомого твірного поліному, для формування систематичних кодів БЧХ можна використовувати відомий алгоритм формування систематичних циклічних кодів, описаний у підрозділі 2.5.5. Тобто, для формування систематичного коду БЧХ необхідно виконати наступні дії.

1. Помножити поліном, який описує вхідне слово, на x^m, де m – степінь твірного поліному, яка визначається через співвідношення (3.25). В результаті отримуємо поліном $A(x)$:

$$A(x) = x^m \cdot P(x). \tag{3.128}$$

2. Розділити отриманий поліном $A(x)$ на твірний поліном $G(x)$ та отримати частку $C(x)$ та остачу від ділення $R(x)$:

$$C(x) = \left[\frac{A(x)}{G(x)}\right]; \qquad R(x) = \frac{A(x)}{G(x)} - C(x). \tag{3.129}$$

3. Формування кодової послідовності $S(x)$ як суми поліномів $A(x)$ та $R(x)$, тобто:

$$S(x) = A(x) + R(x). \tag{3.130}$$

Розглянемо спосіб формування кодів БЧХ, які виправляють подвійні та потрійні помилки, а також способи їхнього декодування з використанням алгоритму Берлекемпа – Мессі та відповідні цифрові електронні декодувальні схеми [56, 57].

Розглянемо код БЧХ DEC, тобто код, який виявляє та виправляє подвійні помилки. Згідно із співвідношенням (3.25) будемо шукати твірний поліном як добуток непарних мінімальних поліномів, враховуючи, що ці поліноми не повинні мати спільного дільника. Будемо шукати корені початкового незвідного поліному $x^4 + x + 1$ як степені примітивного елементу α у поля Галуа $GF(2)$. Відповідно, у поліноміальній формі можна записати [56, 57]:

$$\alpha \leftrightarrow x^4 + x + 1; \qquad \alpha^2 \leftrightarrow x^4 + x + 1; \qquad (3.131)$$

$$\alpha^3 \leftrightarrow x^4 + x^3 + x^2 + x + 1; \qquad \alpha^4 \leftrightarrow x^4 + x + 1.$$

Тепер необхідно знайти породжувальний поліном $G(x)$ як добуток поліномів $x^4 + x + 1$ та $x^4 + x^3 + x^2 + x + 1$:

$$G(x) = (x^4 + x + 1)\cdot(x^4 + x^3 + x^2 + x + 1) = x^8 + x^7 + x^6 + x^4 + 1. \quad (3.132)$$

Будемо вважати, що за умови $m = 8$ кількість інформаційних розрядів у коді складає $k = 7$, тобто, загальна кількість розрядів коду $n = 15$. Узагальнена структура такого систематичного коду показана в таблиці 3.4. Принципова електрична схема кодера, який формує код БЧХ (15, 7), наведена на рис. 3.17.

Таблиця 3.4 – Структура систематичного коду БЧХ (15, 7), який виправляє подвійні помилки

Інформаційні розряди, 7 бітів							Контрольні розряди, 8 бітів							
a_6	a_5	a_4	a_3	a_2	a_1	a_0	c_7	c_6	c_5	c_4	c_3	c_2	c_1	c_0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Номери розрядів														

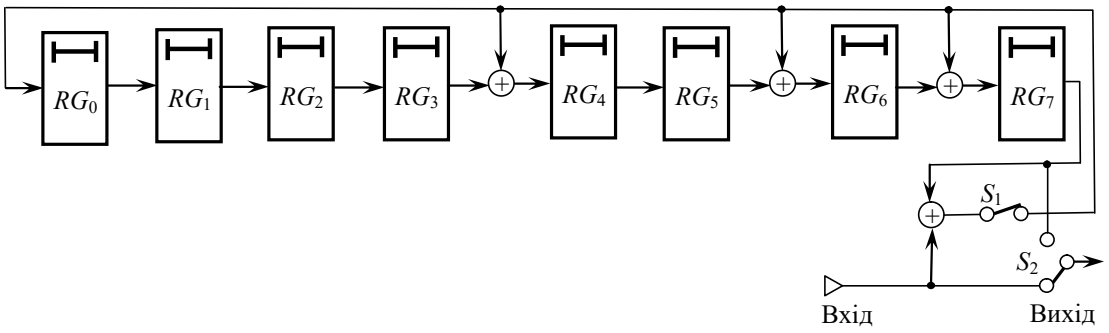


Рис. 3.17 Схема кодування для коду БЧХ (15, 7)

Як видно із таблиці 3.4, у даному випадку для коду БЧХ використаний зворотний порядок нумерації символів. Це є важливим для подальших алгебраїчних операцій, пов'язаних із формуванням контрольних сум.

Будемо формувати контрольні суми як степені, з нульової по чотирнадцяту, примітивних елементів α та α^3. Для елементу α маємо прості результати: α^{14}, α^{13}, α^{12}, α^{11}, α^{10}, α^9, α^8, α^7, α^6, α^5, α^4, α^3, α^2, α^1, α^0.

Відповідно, для степенів елементу α^3 можна записати [52, 53]:

$$\left(\alpha^{14}\right)^3 = \alpha^{42} = \alpha^{42-15\cdot 2} = \alpha^{12}; \qquad \left(\alpha^{13}\right)^3 = \alpha^{39} = \alpha^{39-15\cdot 2} = \alpha^{9};$$

$$\left(\alpha^{13}\right)^3 = \alpha^{39} = \alpha^{39-15\cdot 2} = \alpha^{9}; \qquad \left(\alpha^{12}\right)^3 = \alpha^{36} = \alpha^{36-15\cdot 2} = \alpha^{6};$$

$$\left(\alpha^{11}\right)^3 = \alpha^{33} = \alpha^{33-15\cdot 2} = \alpha^{6}; \qquad \left(\alpha^{10}\right)^3 = \alpha^{30} = \alpha^{30-15\cdot 2} = \alpha^{0}; \quad (3.133)$$

$$\left(\alpha^{9}\right)^3 = \alpha^{27} = \alpha^{27-15} = \alpha^{12}; \qquad \left(\alpha^{8}\right)^3 = \alpha^{24} = \alpha^{24-15} = \alpha^{9};$$

$$\left(\alpha^{7}\right)^3 = \alpha^{21} = \alpha^{21-15} = \alpha^{6}; \qquad \left(\alpha^{6}\right)^3 = \alpha^{18} = \alpha^{18-15} = \alpha^{3};$$

$$\left(\alpha^{5}\right)^3 = \alpha^{15} = \alpha^{15-15} = \alpha^{0}; \qquad \left(\alpha^{4}\right)^3 = \alpha^{12}; \left(\alpha^{3}\right)^3 = \alpha^{9};$$

$$\left(\alpha^{2}\right)^3 = \alpha^{6}; \left(\alpha^{1}\right)^3 = \alpha^{3}; \ \left(\alpha^{0}\right)^3 = \alpha^{0}.$$

Враховуючи співвідношення (3.133), контрольні суми можна сформувати у вигляді, поданому у таблиці 3.5.

Тоді перевірочну матрицю **H** можна записати наступним чином [52, 53]:

$$\mathbf{H} = \begin{bmatrix} \alpha^{12} & \alpha^{9} & \alpha^{6} & \alpha^{3} & \alpha^{0} & \alpha^{12} & \alpha^{9} & \alpha^{6} & \alpha^{3} & \alpha^{0} & \alpha^{12} & \alpha^{9} & \alpha^{6} & \alpha^{3} & \alpha^{0} \\ \alpha^{14} & \alpha^{13} & \alpha^{12} & \alpha^{11} & \alpha^{10} & \alpha^{9} & \alpha^{8} & \alpha^{7} & \alpha^{6} & \alpha^{5} & \alpha^{4} & \alpha^{3} & \alpha^{2} & \alpha^{1} & \alpha^{0} \end{bmatrix}, \quad (3.134)$$

або, через двійкове подання елементів групи Галуа α^n, яке було наведене у таблиці 3.1 [52, 53]:

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (3.135)$$

Таблиця 3.5 – Синдроми помилок для систематичного коду БЧХ (15, 7)

Синдроми		Помилки, які відповідають синдромам
S_0	S_1	
0	0	Відсутність помилки
$\left(\alpha^{14}\right)^3 = \alpha^{12}$	α^{14}	Помилка у першому біті
$\left(\alpha^{13}\right)^3 = \alpha^{9}$	α^{13}	Помилка у другому біті
$\left(\alpha^{12}\right)^3 = \alpha^{6}$	α^{12}	Помилка у третьому біті
$\left(\alpha^{11}\right)^3 = \alpha^{3}$	α^{11}	Помилка у четвертому біті
$\left(\alpha^{10}\right)^3 = \alpha^{0}$	α^{10}	Помилка у п’ятому біті
$\left(\alpha^{9}\right)^3 = \alpha^{12}$	α^{9}	Помилка у шостому біті
$\left(\alpha^{8}\right)^3 = \alpha^{9}$	α^{8}	Помилка у сьомому біті
$\left(\alpha^{7}\right)^3 = \alpha^{6}$	α^{7}	Помилка у восьмому біті
$\left(\alpha^{6}\right)^3 = \alpha^{3}$	α^{6}	Помилка у дев’ятому біті
$\left(\alpha^{5}\right)^3 = \alpha^{0}$	α^{5}	Помилка у десятому біті
$\left(\alpha^{4}\right)^3 = \alpha^{12}$	α^{4}	Помилка в одинадцятому біті
$\left(\alpha^{3}\right)^3 = \alpha^{9}$	α^{3}	Помилка у дванадцятому біті
$\left(\alpha^{2}\right)^3 = \alpha^{6}$	α^{2}	Помилка у тринадцятому біті
$\left(\alpha^{1}\right)^3 = \alpha^{3}$	α^{1}	Помилка у чотирнадцятому біті

$\left(\alpha^0\right)^3 = \alpha^0$	α^0	Помилка у п’ятнадцятому біті

Визначимо вектор помилки **E**, який має 15 компонент:

$$\mathbf{E} = [e_{14}\ e_{13}\ e_{12}\ e_{11}\ e_{10}\ e_9\ e_8\ e_7\ e_6\ e_5\ e_4\ e_3\ e_2\ e_1\ e_0]. \tag{3.136}$$

Нехай аналізується код БЧХ (15, 7), який виправляє подвійні помилки, лише два елементи вектора **E**, визначеного співвідношенням (3.136), можуть мати значення 1, а решта дорівнюють 0. Тоді рівняння синдромів, згідно із узагальненим співвідношенням (3.43), можна записати у вигляді [52, 53]:

$$\mathbf{S} = \begin{pmatrix} S_1 \\ S_0 \end{pmatrix} = \mathbf{H} \cdot \mathbf{E_T} = \begin{pmatrix} \left(\alpha^{14}\right)^3 & \left(\alpha^{13}\right)^3 & \dots & \left(\alpha^{0}\right)^3 \\ \alpha^{14} & \alpha^{13} & \dots & \alpha^{0} \end{pmatrix} \cdot \mathbf{E_T} =$$

$$= \begin{pmatrix} \alpha^{12} & \alpha^{9} & \dots & \alpha^{0} \\ \alpha^{14} & \alpha^{13} & \dots & \alpha^{0} \end{pmatrix} \cdot \mathbf{E_T}. \tag{3.137}$$

З урахуванням співвідношення (3.137), перепишемо рівняння для синдромів помилок у вигляді:

$$S_1 = \alpha^{12} \cdot e_{14} + \alpha^{9} \cdot e_{13} + \ldots + \alpha^{3} \cdot e_1 + \alpha^{0} \cdot e_0;$$

$$S_0 = \alpha^{14} \cdot e_{14} + \alpha^{13} \cdot e_{13} + \ldots + \alpha^{1} \cdot e_1 + \alpha^{0} \cdot e_0. \tag{3.138}$$

Тепер, враховуючи формулу (3.93), рівняння для поліному локаторів помилок $\Lambda(x)$ через степені елементу поля Галуа α можна записати у вигляді [52, 53]:

$$\Lambda(x) = \left(x - \alpha^i\right) \cdot \left(x - \alpha^j\right) = x^2 + A \cdot x + B = x^2 + S_0 \cdot x + \left(S_0^2 + \frac{S_1}{S_0}\right). \tag{3.139}$$

Знайти корені поліному (3.139) можна за описаним вище алгоритмом процедури Ченя, послідовно підставляючи в нього елементи поля Галуа $GF(2^4)$.

Узагальнена структурна схема алгоритму декодування коду БЧХ (15, 7) наведено на рис. 3.18 [52, 53].

Розглянемо приклад пошуку помилок у коді БЧХ (15, 7).

Приклад 3.17. З використанням алгоритму, який наведений на рис. 3.17, знайти помилки у послідовності коду БЧХ (15, 7), вважаючи, що вони виникли у восьмому та дванадцятому бітах.

Для прикладу, який розглядається, згідно із співвідношенням (3.136) вектор помилок **E** записуються у вигляді:

$$\mathbf{E} = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0],$$

тобто, $e_7 = 1$ та $e_3 = 1$, а решта бітів вектора помилки дорівнюють 0.

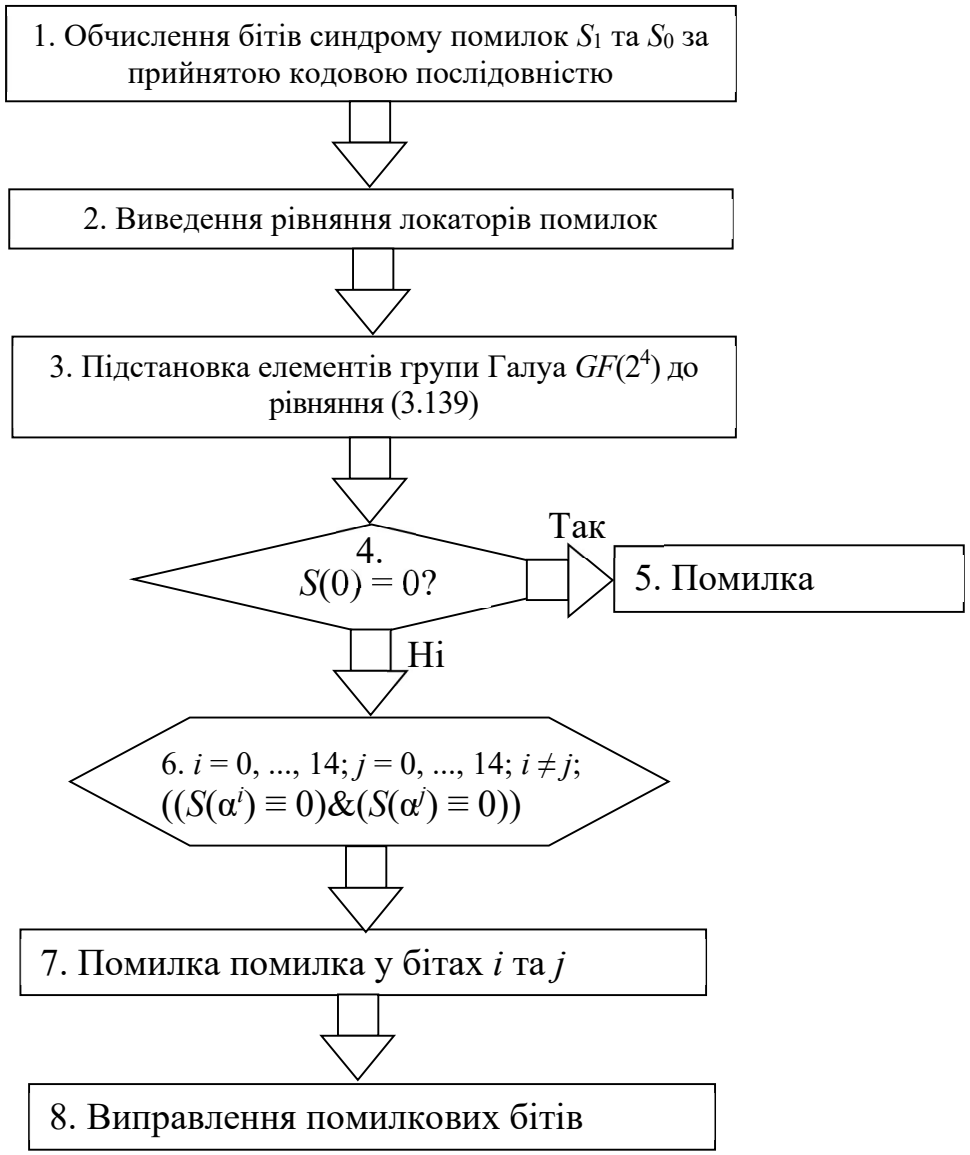


Рис. 3.18 Алгоритм декодування коду БЧХ (15, 7)

Тоді, згідно із таблицею 3.5, можна записати рівняння синдромів помилок для елементу поля Галуа α:

$$S(\alpha^7) = 0; \quad S(\alpha^3) = 0. \tag{3.140}$$

З урахуванням рівняння (3.139), можна записати:

$$\left(\alpha^7\right)^2 + A \cdot \left(\alpha^7\right) + B = 0; \ \left(\alpha^3\right)^2 + A \cdot \left(\alpha^3\right) + B = 0. \tag{3.141}$$

З іншого боку, можна записати:

$$S_0^2 = \left(\alpha^7 + \alpha^3\right)^2 = \left(\alpha^7\right)^2 + \left(\alpha^3\right)^2. \tag{3.142}$$

Сумуючи рівняння (3.141), отримуємо:

$$\left(\left(\alpha^7\right)^2 + \left(\alpha^3\right)^2\right) + A \cdot \left(\alpha^7 + \alpha^3\right) + 2 \cdot B = 0. \tag{3.143}$$

Тоді, відповідно до формули (3.142), враховуючи (3.143), отримуємо лінійне рівняння відносно змінної A:

$$S_0^2 + A \cdot S_0 = 0,$$

яке має простий розв'язок $A = S_0$.

Для визначення другого коефіцієнту B помножимо перше рівняння системи (3.141) на α^7, а друге – на α^3 та просумуємо результати:

$$\left(\left(\alpha^7\right)^3 + A \cdot \left(\alpha^7\right)^2 + B \cdot \alpha^7\right) + \left(\left(\alpha^3\right)^3 + A \cdot \left(\alpha^3\right)^2 + B \cdot \alpha^3\right) =$$

$$= \left(\left(\alpha^7\right)^3 + \left(\alpha^3\right)^3\right) + A \cdot \left(\left(\alpha^7\right)^2 + \left(\alpha^3\right)^2\right) + B \cdot \left(\alpha^7 + \alpha^3\right) = 0. \tag{3.144}$$

Із (3.144), враховуючи (3.141) та (3.142), отримуємо вираз для коефіцієнта B через синдроми помилок та коефіцієнт A [52, 53]:

$$S_1 + A \cdot S_0^2 + B \cdot S_0 = 0. \tag{3.145}$$

Із (3.145), враховуючи тотожність $A = S_0$, остаточно маємо:

$$S_1 + S_0^3 + B \cdot S_0 = 0 \Rightarrow \ B = \frac{S_1 + S_0^3}{S_0} = S_0^2 + \frac{S_1}{S_0}. \tag{3.146}$$

Із отриманого співвідношення (3.146) безпосередньо випливає рівняння (3.139).

Рівняння (3.146) можна розкласти на множники.

$$S_1 = \left(\alpha^7\right)^3 + \left(\alpha^3\right)^3 = \left(\alpha^7 + \alpha^3\right) \cdot \left(\left(\alpha^7\right)^2 + \alpha^7 \cdot \alpha^3 + \left(\alpha^3\right)^2\right). \quad (3.147)$$

$$S_0^2 + \frac{S_1}{S_0} = \left(\alpha^7\right)^2 + \left(\alpha^3\right)^2 + \left(\left(\alpha^7\right)^2 + \alpha^7 \cdot \alpha^3 + \left(\alpha^3\right)^2\right) = \alpha^7 \cdot \alpha^3.$$

Із (3.147), враховуючи, що $S_0 = \alpha^7 + \alpha^3$, рівняння (3.139) переписується у вигляді:

$$\Lambda(x) = x^2 + \left(\alpha^7 + \alpha^3\right) \cdot x + \alpha^7 \cdot \alpha^3 = \left(x + \alpha^7\right) \cdot \left(x + \alpha^3\right). \quad (3.148)$$

Тобто, $\Lambda\left(\alpha^7\right) = 0$ та $\Lambda\left(\alpha^3\right) = 0$, що дійсно відповідає розташуванню помилкових розрядів.

Схема декодеру для коду БЧХ (15, 7), основана на узагальнених схемах, наведених у підрозділі 3.3.5, наведена на рис. 3.19 [52, 53]. Відповідні синдроми помилок для різної відстані між помилковими бітами наведені у таблиці 3.6 [52, 53].

Таблиця 3.6 – Взаємозв'язок моделей помилок із їхніми синдромами

Модель відстані помилок	Значення моделей відстаней помилок	Конфігурації синдромів, які визначають моделі відстаней помилок							
		RG_0	RG_1	RG_2	RG_3	RG_4	RG_5	RG_6	RG_7
(15)	$R(\alpha) = 1$	1	0	0	0	0	0	0	0
(1,14)	$R(\alpha) = 1 + \alpha$	1	1	0	0	0	0	0	0
(2,13)	$R(\alpha) = 1 + \alpha^2$	1	0	1	0	0	0	0	0
(3,12)	$R(\alpha) = 1 + \alpha^3$	1	0	0	1	0	0	0	0
(4,11)	$R(\alpha) = 1 + \alpha^4$	1	0	0	0	1	0	0	0
(5,10)	$R(\alpha) = 1 + \alpha^5$	1	0	0	0	0	1	0	0
(6,9)	$R(\alpha) = 1 + \alpha^6$	1	0	0	0	0	0	1	0
(7,8)	$R(\alpha) = 1 + \alpha^7$	1	0	0	0	0	0	0	1

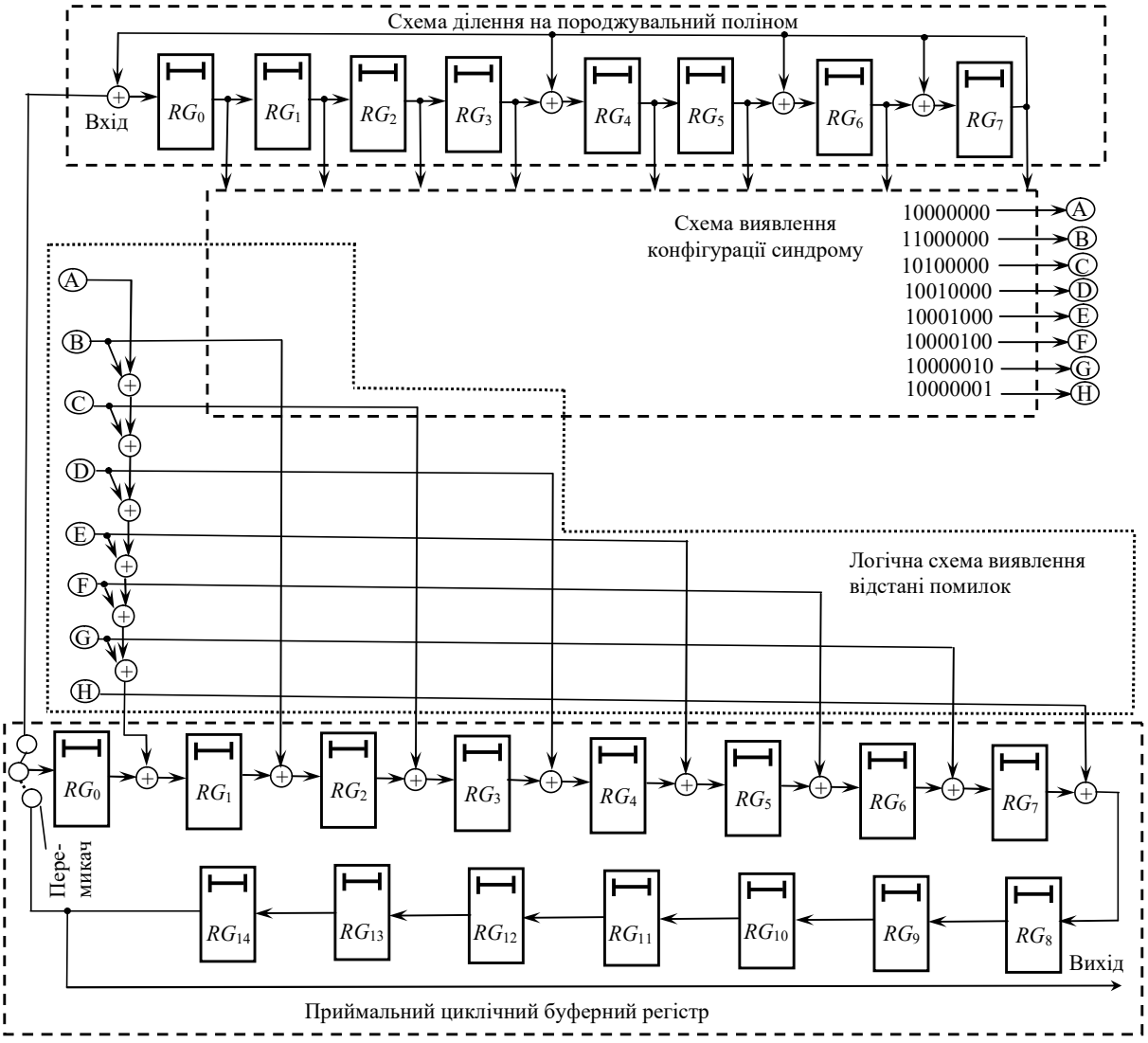


Рис. 3.19 Принципова електрична схема декодера для коду БЧХ (15, 7)

Розглянемо процес виправлення подвійної помилки у коді БЧХ (15, 7) з використанням логічної схеми, яка наведена на рис. 3.18, на конкретному прикладі [52, 53].

Приклад 3.18. Проаналізувати роботу логічної схеми, наведеної на рис. 3.18, для випадку наявності у коді БЧХ (15, 7) подвійної помилки у четвертому та восьмому розрядах.

Значення буферних регістрів приймального пристрою RG_0 – RG_{14} на різних тактах роботи декодувальної схеми показано у таблиці 3.7. Із даних, наведених у таблиці 3.7, видно, що до десятого такту здійснюється циклічний зсув прийнятих бітів, а на дев'ятому такті формується синдром помилки і значення бітів коду $\overline{c_7}$ та $\overline{a_3}$ автоматичне змінюється на протилежне.

Таблиця 3.7 – Значення регістрів $RG_0 – RG_{14}$ приймальної схеми декодера

№	RG_0	RG_1	RG_2	RG_3	RG_4	RG_5	RG_6	RG_7	RG_8	RG_9	RG_{10}	RG_{11}	RG_{12}	RG_{13}	RG_{14}
1.	c_0	c_1	c_2	c_3	c_4	c_5	c_6	$\overline{c_7}$	a_0	a_1	a_2	$\overline{a_3}$	a_4	a_5	a_6
2.	a_6	c_0	c_1	c_2	c_3	c_4	c_5	c_6	$\overline{c_7}$	a_0	a_1	a_2	$\overline{a_3}$	a_4	a_5
3.	a_5	a_6	c_0	c_1	c_2	c_3	c_4	c_5	c_6	$\overline{c_7}$	a_0	a_1	a_2	$\overline{a_3}$	a_4
4.	a_4	a_5	a_6	c_0	c_1	c_2	c_3	c_4	c_5	c_6	$\overline{c_7}$	a_0	a_1	a_2	$\overline{a_3}$
5.	$\overline{a_3}$	a_4	a_5	a_6	c_0	c_1	c_2	c_3	c_4	c_5	c_6	$\overline{c_7}$	a_0	a_1	a_2
6.	a_2	$\overline{a_3}$	a_4	a_5	a_6	c_0	c_1	c_2	c_3	c_4	c_5	c_6	$\overline{c_7}$	a_0	a_1
7.	a_1	a_2	$\overline{a_3}$	a_4	a_5	a_6	c_0	c_1	c_2	c_3	c_4	c_5	c_6	$\overline{c_7}$	a_0
8.	a_0	a_1	a_2	$\overline{a_3}$	a_4	a_5	a_6	c_0	c_1	c_2	c_3	c_4	c_5	c_6	$\overline{c_7}$
9.	$\overline{c_7}$	a_0	a_1	a_2	$\overline{a_3}$	a_4	a_5	a_6	c_0	c_1	c_2	c_3	c_4	c_5	c_6
10.	c_6	c_7	a_0	a_1	a_2	a_3	a_4	a_5	a_6	c_0	c_1	c_2	c_3	c_4	c_5
11.	c_5	c_6	c_7	a_0	a_1	a_2	a_3	a_4	a_5	a_6	c_0	c_1	c_2	c_3	c_4
12.	c_4	c_5	c_6	c_7	a_0	a_1	a_2	a_3	a_4	a_5	a_6	c_0	c_1	c_2	c_3
13.	c_3	c_4	c_5	c_6	c_7	a_0	a_1	a_2	a_3	a_4	a_5	a_6	c_0	c_1	c_2
14.	c_2	c_3	c_4	c_5	c_6	c_7	a_0	a_1	a_2	a_3	a_4	a_5	a_6	c_0	c_1
15.	c_1	c_2	c_3	c_4	c_5	c_6	c_7	a_0	a_1	a_2	a_3	a_4	a_5	a_6	c_0
16.	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	a_0	a_1	a_2	a_3	a_4	a_5	a_6
17.	–	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	a_0	a_1	a_2	a_3	a_4	a_5
18.	–	–	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	a_0	a_1	a_2	a_3	a_4
19.	–	–	–	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	a_0	a_1	a_2	a_3
20.	–	–	–	–	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	a_0	a_1	a_2
21.	–	–	–	–	–	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	a_0	a_1
22.	–	–	–	–	–	–	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	a_0
23.	–	–	–	–	–	–	–	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7
24.	–	–	–	–	–	–	–	–	c_0	c_1	c_2	c_3	c_4	c_5	c_6
25.	–	–	–	–	–	–	–	–	–	c_0	c_1	c_2	c_3	c_4	c_5
26.	–	–	–	–	–	–	–	–	–	–	c_0	c_1	c_2	c_3	c_4
27.	–	–	–	–	–	–	–	–	–	–	–	c_0	c_1	c_2	c_3
28.	–	–	–	–	–	–	–	–	–	–	–	–	c_0	c_1	c_2
29.	–	–	–	–	–	–	–	–	–	–	–	–	–	c_0	c_1
30.	–	–	–	–	–	–	–	–	–	–	–	–	–	–	c_0

Тоді, починаючи з десятого такту, перемикач переходить із верхнього до нижнього положення, і починається послідовне виведення бітів виправленої кодової послідовності із регістру RG_{14}.

Оскільки порядок породжувального поліному дорівнює 8, але кодова комбінація має 15 розрядів, запишемо спочатку остачу від ділення на породжувальний поліном у загальному вигляді [52, 53]:

$$R(\alpha) = e_{14} \cdot \alpha^{14} + e_{13} \cdot \alpha^{13} + e_{12} \cdot \alpha^{12} + \overline{e_{11}} \cdot \alpha^{11} + e_{10} \cdot \alpha^{10} + e_9 \cdot \alpha^9 + \\ + e_8 \cdot \alpha^8 + \overline{e_7} \cdot \alpha^7 + e_6 \cdot \alpha^6 + e_5 \cdot \alpha^5 + e_4 \cdot \alpha^4 + e_3 \cdot \alpha^3 + e_2 \cdot \alpha^2 + \\ + e_1 \cdot \alpha^1 + e_0 . \quad (3.149)$$

Згідно із теорією циклотомічних класів, розглянутою у підрозділі 3.6.10 першої частини посібника, можна виразити у співвідношенні (3.149) вищі степені примітивного елементу α, з восьмої по чотирнадцяту, через його нижчі степені. Відповідно маємо [52, 53]:

$$\alpha^8 = \alpha^7 \cdot \alpha = \alpha^7 + \alpha^6 + \alpha^4 + 1;$$

$$\alpha^9 = \alpha^8 \cdot \alpha = \alpha^8 + \alpha^7 + \alpha^5 + \alpha = \left(\alpha^7 + \alpha^6 + \alpha^4 + 1\right) + \alpha^7 + \alpha^5 + \alpha = \\ = \alpha^6 + \alpha^5 + \alpha^4 + \alpha + 1;$$

$$\alpha^{10} = \alpha^9 \cdot \alpha = \alpha \cdot \left(\alpha^6 + \alpha^5 + \alpha^4 + \alpha + 1\right) = \alpha^7 + \alpha^6 + \alpha^5 + \alpha^2 + \alpha;$$

$$\alpha^{11} = \alpha^{10} \cdot \alpha = \alpha \cdot \left(\alpha^7 + \alpha^6 + \alpha^5 + \alpha^2 + \alpha\right) = \alpha^8 + \alpha^7 + \alpha^6 + \alpha^3 + \alpha^2 = \\ = \left(\alpha^7 + \alpha^6 + \alpha^4 + 1\right) + \alpha^7 + \alpha^6 + \alpha^3 + \alpha^2 = \alpha^4 + \alpha^3 + \alpha^2 + 1; \quad (3.150)$$

$$\alpha^{12} = \alpha^{11} \cdot \alpha = \alpha \cdot \left(\alpha^4 + \alpha^3 + \alpha^2 + 1\right) = \alpha^5 + \alpha^4 + \alpha^3 + \alpha;$$

$$\alpha^{13} = \alpha^{12} \cdot \alpha = \alpha \cdot \left(\alpha^5 + \alpha^4 + \alpha^3 + \alpha\right) = \alpha^6 + \alpha^5 + \alpha^4 + \alpha^2;$$

$$\alpha^{14} = \alpha^{13} \cdot \alpha = \alpha \cdot \left(\alpha^6 + \alpha^5 + \alpha^4 + \alpha^2\right) = \alpha^7 + \alpha^6 + \alpha^5 + \alpha^3 .$$

З урахуванням (3.150), співвідношення (3.149) можна переписати у вигляді [52, 53]:

$$R(\alpha) = \left(e_{14} + e_{10} + e_8 + \overline{e_7}\right) \cdot \alpha^7 + \left(e_{14} + e_{13} + e_{10} + e_9 + e_8 + e_6\right) \cdot \alpha^6 + \\ + \left(e_{14} + e_{13} + e_{12} + e_{10} + e_9 + e_5\right) \cdot \alpha^5 + \left(e_{13} + e_{12} + \overline{e_{11}} + e_9 + e_8 + e_4\right) \cdot \alpha^4 + \\ + \left(e_{14} + e_{12} + \overline{e_{11}} + e_3\right) \cdot \alpha^3 + \left(e_{13} + \overline{e_{11}} + e_{10} + e_2\right) \cdot \alpha^2 + \\ + \left(e_{12} + e_{10} + e_9 + e_1\right) \cdot \alpha + \left(\overline{e_{11}} + e_9 + e_8 + e_0\right). \quad (3.151)$$

Враховуючи, що всі коефіцієнти e, крім $\overline{e_7}$ та $\overline{e_{11}}$ дорівнюють 0, з урахуванням (3.151), остаточно маємо:

$$R(\alpha) = \overline{(e_7)} \cdot \alpha^7 + \overline{(e_{11})} \cdot \alpha^4 + \overline{(e_{11})} \cdot \alpha^3 + \overline{(e_{11})} \cdot \alpha^2 + \overline{(e_{11})} = 10011101 \ ,$$

що для прикладу, який розглядається, і є синдромом визначеної подвійної помилки.

У розглянутому прикладі для пошуку помилки у коді БЧХ використовується узагальнений метод аналізу кодових послідовностей, пов'язаний із пошуком синдрому помилки через аналіз остач. Особливості використання інших методів пошуку помилок у кодах БЧХ, описаних у підрозділі 3.3.4, будуть розглянуті у підрозділах 3.4.6 та 3.4.7 для декодування посілдовностей кодів Ріда – Соломона, які можна вважати різновидом кодів БЧХ.

3.3.6.2 Формування та декодування систематичних кодів Боуза – Чоудхурі – Хоквінгема, які виправляють потрійні помилки

Зрозуміло, для збільшення кількості помилок, які може виправити код БЧХ, необхідно збільшувати кількість контрольних бітів. Проте, як і для кодів Хеммінга та циклічних кодів, розглянутих у другому підрозділі, за умови зростання довжини блоку надлишковість кодів БЧХ стає меншою [52, 53].

Розглянемо тепер код БЧХ (15, 5), який виправляє потрійні помилки. помилки. Цей код використовується в системах цифрового магнітного запису звукової інформації [52, 53].

Породжувальним поліномом для коду БЧХ (15, 5) є поліном [52, 53]:

$$g(x) = x^{10} + x^8 + x^5 + x^4 + x^2 + x + 1. \qquad (3.152)$$

Відповідний результат був отриманий у прикладі 3.5, співвідношення (3.27).

Головними параметрами коду БЧХ (15, 5) є наступні [52, 53]:

1. Загальна кількість розрядів коду $n = 15$.

2. Кількість контрольних розрядів коду $r = 10$.

3. Кількість інформаційних розрядів коду $k = 5$.

Узагальнена структура систематичного коду БЧХ (15, 5) ТЕС показана у таблиці 3.8, а принципова електрична схема кодера, який формує такий код, наведена на рис. 3.20 [52, 53].

Таблиця 3.8 – Структура систематичного коду БЧХ (15, 7), який виправляє подвійні помилки

Інформаційні розряди, 5 бітів					Контрольні розряди, 10 бітів									
a_4	a_3	a_2	a_1	a_0	c_9	c_8	c_7	c_6	c_5	c_4	c_3	c_2	c_1	c_0
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Номери розрядів														

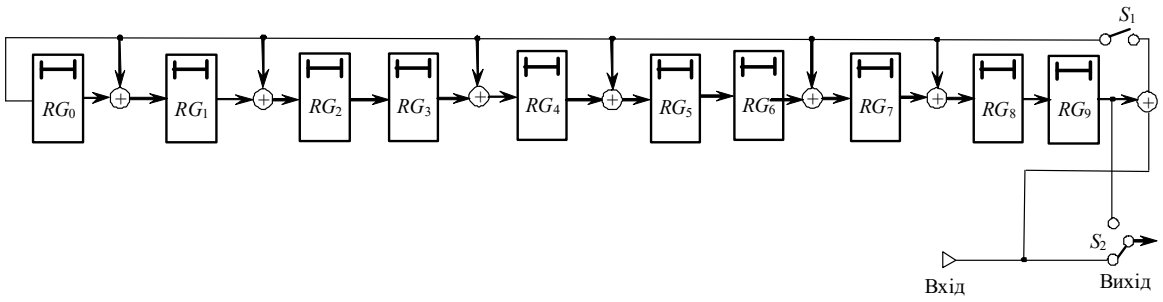


Рис. 3.20 Схема кодування для коду БЧХ (15, 5)

Розглянемо окремо моделі відстаней помилок для однієї, подвійної та потрійної помилки [52, 53].

Для однієї помилки – модель відстані помилок (15).

Для подвійної помилки – моделі відстаней помилок (1, 14), (2, 13), (3, 12), (4, 11), (5, 10), (6, 9), (7, 8).

У разі потрійної помилки моделі відстані мають вигляд (i, j, k). Задамо умови $i + j + k = 15$, $i \le j$, $i \le k$, $j < k$. Тоді необхідно окремо розглянути випадки $i = 1$, $i = 2$, $i = 3$, $i = 4$ та $i = 5$.

Відповідно, для $i = 1$:

(1, 1, 13), (1, 2, 12), (1, 3, 11), (1, 4, 10), (1, 5, 9), (1, 6, 8), (1, 7, 7),

(1, 8, 6), (1, 9, 5), (1, 10, 4), (1, 11, 3), (1, 12, 2), (1, 13, 7).

Для $i = 2$:

(2, 2, 11), (2, 3, 10), (2, 4, 9), (2, 5, 8), (2, 6, 7),

(2, 7, 6), (2, 8, 5), (2, 9, 4), (2, 10, 3).

Для $i = 3$:

(3, 3, 9), (3, 4, 8), (3, 5, 7), (3, 6, 6), (3, 7, 5), (3, 8, 4).

Для $i = 4$:

(4, 4, 7), (4, 5, 6), (4, 6, 5).

Для $i = 5$:

(5, 5, 5).

Загальна кількість моделей відстаней помилок дорівнює 31. Конфігурації помилкових потрійних помилок наведені у таблиці 3.9 [52, 53].

Таблиця 3.9 – Конфігурація потрійних помилок для коду БЧХ (15, 5)

Модель відстаней помилок	Конфігурація відстаней потрійних помилок				
(1, 1, 13)	(e_2, e_1, e_0) (e_7, e_6, e_5) (e_{12}, e_{11}, e_{10})	(e_3, e_2, e_1) (e_8, e_7, e_6) (e_{13}, e_{12}, e_{11})	(e_4, e_3, e_2) (e_9, e_8, e_7) (e_{14}, e_{13}, e_{12})	(e_5, e_4, e_3) (e_{10}, e_9, e_8) (e_{14}, e_{13}, e_0)	(e_6, e_5, e_4) (e_{11}, e_{10}, e_9) (e_{14}, e_1, e_0)
(1, 2, 12)	(e_3, e_1, e_0) (e_9, e_6, e_5) (e_{13}, e_{11}, e_{10})	(e_4, e_2, e_1) (e_9, e_7, e_6) (e_{14}, e_{12}, e_{11})	(e_5, e_3, e_2) (e_{10}, e_8, e_7) (e_{13}, e_{12}, e_0)	(e_6, e_4, e_3) (e_{11}, e_9, e_8) (e_{14}, e_{13}, e_1)	(e_7, e_5, e_4) (e_{12}, e_{10}, e_9) (e_{14}, e_2, e_0)
(1, 3, 11)	(e_4, e_1, e_0) (e_9, e_6, e_5) (e_{14}, e_{11}, e_{10})	(e_5, e_2, e_1) (e_{10}, e_7, e_6) (e_{12}, e_{11}, e_0)	(e_6, e_3, e_2) (e_{11}, e_9, e_7) (e_{13}, e_{12}, e_1)	(e_7, e_4, e_3) (e_{12}, e_9, e_8) (e_{14}, e_{13}, e_2)	(e_8, e_5, e_4) (e_{13}, e_{10}, e_9) (e_{14}, e_3, e_0)
(1, 4, 10)	(e_5, e_1, e_0) (e_{10}, e_6, e_5) (e_{11}, e_{11}, e_{10})	(e_6, e_2, e_1) (e_{11}, e_7, e_6) (e_{12}, e_{11}, e_2)	(e_7, e_3, e_2) (e_{12}, e_9, e_7) (e_{13}, e_{12}, e_3)	(e_9, e_4, e_3) (e_{13}, e_9, e_8) (e_{14}, e_{13}, e_4)	(e_9, e_5, e_4) (e_{14}, e_{10}, e_9) (e_{14}, e_5, e_0)
(1, 5, 9)	(e_6, e_1, e_0) (e_{11}, e_6, e_5) (e_{11}, e_{10}, e_1)	(e_7, e_2, e_1) (e_{12}, e_7, e_6) (e_{12}, e_{11}, e_2)	(e_8, e_3, e_2) (e_{13}, e_8, e_7) (e_{13}, e_{12}, e_3)	(e_9, e_4, e_3) (e_{14}, e_9, e_8) (e_{14}, e_{13}, e_4)	(e_{10}, e_5, e_4) (e_{10}, e_9, e_0) (e_{14}, e_5, e_0)
(1, 6, 8)	(e_7, e_1, e_0) (e_{12}, e_6, e_5) (e_{11}, e_{10}, e_2)	(e_8, e_2, e_1) (e_{13}, e_7, e_6) (e_{12}, e_{11}, e_3)	(e_9, e_3, e_2) (e_{14}, e_8, e_7) (e_{14}, e_{12}, e_4)	(e_{10}, e_4, e_3) (e_9, e_8, e_0) (e_{14}, e_{13}, e_5)	(e_{11}, e_5, e_4) (e_{10}, e_9, e_1) (e_{14}, e_6, e_0)
(1, 7, 7)	(e_8, e_1, e_0) (e_{13}, e_6, e_5) (e_{11}, e_{10}, e_3)	(e_9, e_2, e_1) (e_{14}, e_7, e_6) (e_{12}, e_{11}, e_4)	(e_{10}, e_3, e_2) (e_9, e_7, e_0) (e_{13}, e_{12}, e_5)	(e_{11}, e_4, e_3) (e_9, e_8, e_1) (e_{14}, e_{13}, e_6)	(e_{12}, e_5, e_4) (e_{10}, e_9, e_2) (e_{14}, e_7, e_0)

Таблиця 3.9 – Конфігурація потрійних помилок для коду БЧХ (15, 5) (продовження)

Модель відстаней помилок	Конфігурація відстаней потрійних помилок				
(1, 8, 6)	(e_9, e_1, e_0) (e_{14}, e_6, e_5) (e_{11}, e_{10}, e_4)	(e_{10}, e_2, e_1) (e_7, e_6, e_0) (e_{12}, e_{11}, e_5)	(e_{11}, e_3, e_2) (e_8, e_7, e_1) (e_{13}, e_{12}, e_6)	(e_{12}, e_4, e_3) (e_9, e_8, e_2) (e_{14}, e_2, e_7)	(e_{13}, e_5, e_4) (e_{10}, e_9, e_2) (e_{14}, e_9, e_0)
(1, 9, 5)	(e_{10}, e_1, e_0) (e_6, e_5, e_0) (e_{11}, e_{10}, e_5)	(e_{11}, e_2, e_1) (e_7, e_6, e_1) (e_{12}, e_{11}, e_6)	(e_{12}, e_3, e_2) (e_9, e_7, e_2) (e_{13}, e_{12}, e_7)	(e_{13}, e_4, e_3) (e_9, e_8, e_3) (e_{14}, e_{13}, e_8)	(e_{14}, e_5, e_4) (e_{10}, e_9, e_4) (e_{14}, e_{10}, e_0)
(1, 10, 4)	(e_{11}, e_1, e_0) (e_6, e_5, e_1) (e_{11}, e_{10}, e_6)	(e_{12}, e_2, e_1) (e_7, e_6, e_2) (e_{12}, e_{11}, e_7)	(e_{13}, e_3, e_2) (e_9, e_7, e_3) (e_{13}, e_{12}, e_8)	(e_{14}, e_4, e_3) (e_9, e_8, e_4) (e_{14}, e_{13}, e_9)	(e_5, e_4, e_0) (e_{10}, e_9, e_5) (e_{14}, e_{11}, e_0)
(1, 11, 3)	(e_{12}, e_1, e_0) (e_6, e_5, e_2) (e_{11}, e_{10}, e_7)	(e_{13}, e_2, e_1) (e_7, e_6, e_3) (e_{12}, e_{11}, e_8)	(e_{14}, e_3, e_2) (e_8, e_7, e_4) (e_{13}, e_{12}, e_9)	(e_4, e_3, e_0) (e_9, e_8, e_5) (e_{14}, e_{13}, e_{10})	(e_5, e_4, e_1) (e_{10}, e_9, e_6) (e_{14}, e_{11}, e_0)
(1, 12, 2)	(e_{13}, e_1, e_0) (e_6, e_5, e_3) (e_{11}, e_{10}, e_8)	(e_{14}, e_2, e_1) (e_7, e_6, e_4) (e_{12}, e_{11}, e_9)	(e_3, e_2, e_0) (e_8, e_7, e_5) (e_{13}, e_{12}, e_{10})	(e_4, e_4, e_3) (e_9, e_8, e_6) (e_{14}, e_{13}, e_{11})	(e_5, e_4, e_2) (e_{10}, e_9, e_7) (e_{14}, e_7, e_0)
(2, 2, 11)	(e_4, e_2, e_0) (e_9, e_7, e_5) (e_{14}, e_{12}, e_{10})	(e_5, e_3, e_1) (e_{10}, e_9, e_6) (e_{13}, e_{11}, e_0)	(e_6, e_4, e_2) (e_{11}, e_9, e_7) (e_{14}, e_{12}, e_1)	(e_7, e_5, e_3) (e_{12}, e_{10}, e_9) (e_{12}, e_2, e_0)	(e_9, e_6, e_4) (e_{13}, e_{11}, e_9) (e_{14}, e_3, e_1)
(2, 3, 10)	(e_5, e_2, e_0) (e_{10}, e_7, e_5) (e_{12}, e_{10}, e_0)	(e_6, e_3, e_1) (e_{11}, e_9, e_6) (e_{13}, e_{11}, e_1)	(e_7, e_4, e_2) (e_{12}, e_9, e_7) (e_{14}, e_{12}, e_2)	(e_8, e_5, e_3) (e_{13}, e_{10}, e_8) (e_{13}, e_3, e_0)	(e_9, e_6, e_4) (e_{14}, e_{11}, e_9) (e_{14}, e_4, e_1)
(2, 4, 9)	(e_6, e_2, e_0) (e_{11}, e_7, e_3) (e_{12}, e_{10}, e_1)	(e_7, e_3, e_1) (e_{12}, e_8, e_6) (e_{13}, e_{11}, e_2)	(e_9, e_4, e_2) (e_{13}, e_9, e_7) (e_{14}, e_{12}, e_3)	(e_9, e_5, e_3) (e_{14}, e_{10}, e_8) (e_{13}, e_4, e_0)	(e_{10}, e_6, e_4) (e_{11}, e_9, e_0) (e_{14}, e_5, e_1)
(2, 5, 8)	(e_7, e_2, e_0) (e_{12}, e_7, e_5) (e_{12}, e_{10}, e_2)	(e_8, e_3, e_1) (e_{13}, e_8, e_6) (e_{13}, e_{11}, e_3)	(e_9, e_4, e_2) (e_{14}, e_9, e_7) (e_{14}, e_{12}, e_4)	(e_{10}, e_5, e_3) (e_{10}, e_8, e_0) (e_{13}, e_5, e_0)	(e_{11}, e_6, e_4) (e_{11}, e_9, e_1) (e_{14}, e_6, e_1)
(2, 6, 7)	(e_8, e_2, e_0) (e_{13}, e_7, e_5) (e_{12}, e_{10}, e_3)	(e_9, e_3, e_1) (e_{14}, e_9, e_6) (e_{13}, e_{11}, e_4)	(e_{10}, e_4, e_2) (e_9, e_7, e_0) (e_{14}, e_{12}, e_5)	(e_{11}, e_5, e_3) (e_{10}, e_8, e_1) (e_{13}, e_6, e_0)	(e_{12}, e_6, e_4) (e_{11}, e_9, e_1) (e_{14}, e_7, e_1)

Таблиця 3.9 – Конфігурація потрійних помилок для коду БЧХ (15,5) (продовження)

Модель відстаней помилок	Конфігурація відстаней потрійних помилок				
(2, 7, 6)	(e_9, e_2, e_0) (e_{14}, e_7, e_5) (e_{12}, e_{10}, e_4)	(e_{10}, e_3, e_1) (e_9, e_6, e_0) (e_{13}, e_{11}, e_5)	(e_{11}, e_4, e_2) (e_9, e_7, e_1) (e_{14}, e_{12}, e_6)	(e_{12}, e_5, e_3) (e_{10}, e_8, e_2) (e_{13}, e_7, e_0)	(e_{13}, e_6, e_4) (e_{11}, e_9, e_3) (e_{14}, e_8, e_1)
(2, 8, 5)	(e_{10}, e_2, e_0) (e_7, e_5, e_0) (e_{12}, e_{10}, e_5)	(e_{11}, e_3, e_1) (e_8, e_6, e_1) (e_{13}, e_{11}, e_6)	(e_{12}, e_4, e_2) (e_9, e_7, e_2) (e_{14}, e_{12}, e_7)	(e_{13}, e_5, e_3) (e_{10}, e_8, e_3) (e_{13}, e_8, e_0)	(e_{14}, e_6, e_4) (e_{11}, e_9, e_4) (e_{14}, e_9, e_1)
(2, 9, 4)	(e_{11}, e_2, e_0) (e_7, e_5, e_1) (e_{12}, e_{10}, e_6)	(e_{12}, e_3, e_1) (e_8, e_6, e_2) (e_{13}, e_{11}, e_7)	(e_{13}, e_4, e_2) (e_9, e_7, e_3) (e_{14}, e_{12}, e_8)	(e_{14}, e_5, e_3) (e_{10}, e_8, e_4) (e_{13}, e_9, e_0)	(e_6, e_4, e_0) (e_{11}, e_9, e_5) (e_{14}, e_{10}, e_1)
(2, 10, 3)	(e_{12}, e_2, e_0) (e_7, e_5, e_2) (e_{12}, e_{10}, e_7)	(e_{13}, e_3, e_1) (e_8, e_6, e_3) (e_{13}, e_{11}, e_8)	(e_{14}, e_4, e_2) (e_9, e_7, e_4) (e_{14}, e_{12}, e_9)	(e_5, e_3, e_0) (e_{10}, e_8, e_5) (e_{13}, e_{10}, e_0)	(e_6, e_4, e_1) (e_{11}, e_9, e_5) (e_{14}, e_{11}, e_1)
(3, 3, 9)	(e_6, e_3, e_0) (e_{11}, e_9, e_5) (e_{12}, e_{10}, e_1)	(e_7, e_4, e_1) (e_{12}, e_9, e_6) (e_{14}, e_{11}, e_2)	(e_8, e_5, e_2) (e_{13}, e_{10}, e_7) (e_{12}, e_3, e_0)	(e_9, e_6, e_3) (e_{14}, e_{11}, e_8) (e_{13}, e_4, e_1)	(e_{10}, e_7, e_4) (e_{12}, e_9, e_0) (e_{14}, e_5, e_2)
(3, 4, 8)	(e_7, e_3, e_0) (e_{12}, e_9, e_5) (e_{13}, e_{10}, e_2)	(e_8, e_4, e_1) (e_{13}, e_9, e_6) (e_{14}, e_{11}, e_3)	(e_9, e_5, e_2) (e_{14}, e_{10}, e_7) (e_{12}, e_4, e_0)	(e_{10}, e_6, e_3) (e_{11}, e_8, e_0) (e_{13}, e_5, e_1)	(e_{11}, e_7, e_4) (e_{12}, e_9, e_1) (e_{14}, e_6, e_2)
(3, 5, 7)	(e_8, e_3, e_0) (e_{13}, e_9, e_5) (e_{13}, e_{10}, e_3)	(e_9, e_4, e_1) (e_{14}, e_9, e_6) (e_{14}, e_{11}, e_4)	(e_{10}, e_5, e_2) (e_{10}, e_7, e_0) (e_{12}, e_5, e_0)	(e_{11}, e_6, e_3) (e_{11}, e_9, e_0) (e_{13}, e_6, e_1)	(e_{12}, e_7, e_4) (e_{12}, e_9, e_2) (e_{14}, e_7, e_2)
(3, 6, 6)	(e_9, e_3, e_0) (e_{14}, e_9, e_5) (e_{13}, e_{10}, e_4)	(e_{10}, e_4, e_1) (e_9, e_6, e_0) (e_{14}, e_{11}, e_5)	(e_{11}, e_5, e_2) (e_{10}, e_7, e_1) (e_{12}, e_6, e_0)	(e_{12}, e_6, e_3) (e_{11}, e_8, e_2) (e_{13}, e_7, e_1)	(e_{13}, e_7, e_4) (e_{12}, e_9, e_3) (e_{14}, e_9, e_2)
(3, 7, 5)	(e_{10}, e_3, e_0) (e_8, e_5, e_0) (e_{13}, e_{10}, e_5)	(e_{11}, e_4, e_1) (e_9, e_6, e_1) (e_{14}, e_{11}, e_6)	(e_{12}, e_5, e_2) (e_{10}, e_7, e_2) (e_{12}, e_7, e_0)	(e_{13}, e_6, e_3) (e_{11}, e_8, e_3) (e_{13}, e_8, e_1)	(e_{14}, e_7, e_4) (e_{12}, e_9, e_4) (e_{14}, e_9, e_2)

Таблиця 3.9 – Конфігурація потрійних помилок для коду БЧХ (15, 5) (закінчення)

Модель відстаней помилок	Конфігурація відстаней потрійних помилок				
(3, 8, 4)	(e_{11}, e_3, e_0) (e_8, e_5, e_1) (e_{13}, e_{10}, e_6)	(e_{12}, e_4, e_1) (e_9, e_6, e_2) (e_{14}, e_{11}, e_7)	(e_{13}, e_5, e_2) (e_{10}, e_7, e_3) (e_{12}, e_8, e_0)	(e_{14}, e_6, e_3) (e_{11}, e_8, e_4) (e_{13}, e_9, e_1)	(e_7, e_4, e_0) (e_{12}, e_9, e_5) (e_{14}, e_{10}, e_2)
(4, 4, 7)	(e_8, e_4, e_0) (e_{13}, e_9, e_5) (e_{14}, e_{10}, e_3)	(e_9, e_5, e_1) (e_{14}, e_{10}, e_6) (e_{11}, e_4, e_0)	(e_{10}, e_6, e_2) (e_{12}, e_8, e_0) (e_{12}, e_5, e_1)	(e_{11}, e_7, e_3) (e_{12}, e_8, e_1) (e_{13}, e_6, e_2)	(e_{12}, e_8, e_4) (e_{13}, e_9, e_2) (e_{14}, e_8, e_3)
(4, 5, 6)	(e_9, e_4, e_0) (e_{14}, e_9, e_5) (e_{14}, e_{10}, e_4)	(e_{10}, e_5, e_1) (e_{10}, e_6, e_0) (e_{11}, e_5, e_0)	(e_{11}, e_6, e_2) (e_{11}, e_7, e_1) (e_{12}, e_6, e_1)	(e_{12}, e_7, e_3) (e_{12}, e_8, e_2) (e_{13}, e_7, e_2)	(e_{13}, e_8, e_4) (e_{13}, e_9, e_3) (e_{14}, e_8, e_3)
(4, 6, 5)	(e_{10}, e_4, e_0) (e_9, e_5, e_0) (e_{14}, e_{10}, e_5)	(e_{11}, e_5, e_1) (e_{10}, e_6, e_1) (e_{11}, e_6, e_0)	(e_{12}, e_6, e_2) (e_{11}, e_7, e_2) (e_{12}, e_7, e_1)	(e_{13}, e_7, e_3) (e_{12}, e_8, e_3) (e_{13}, e_8, e_2)	(e_{14}, e_8, e_4) (e_{13}, e_9, e_4) (e_{14}, e_9, e_3)
(5, 5, 5)	(e_{10}, e_5, e_0)	(e_{11}, e_6, e_1)	(e_{12}, e_7, e_2)	(e_{13}, e_8, e_3)	(e_{14}, e_9, e_4)

Загалом код БЧХ (15, 5) містить три таких контрольних ряди [52, 53].

1. Контрольний ряд від α, який задається поліномом $x^4 + x + 1$.

2. Контрольний ряд від α^3, який задається поліномом $x^4 + x^3 + x^2 + x + 1$.

3. Контрольний ряд від α^5, який задається поліномом $x^2 + x + 1$.

Матриця контролю записується у вигляді:

$$\mathbf{H} = \begin{pmatrix} \left(\alpha^{14}\right)^5 & \left(\alpha^{13}\right)^5 & \dots & \left(\alpha^{1}\right)^5 & \left(\alpha^{0}\right)^5 \\ \left(\alpha^{14}\right)^3 & \left(\alpha^{13}\right)^3 & \dots & \left(\alpha^{1}\right)^3 & \left(\alpha^{0}\right)^3 \\ \alpha^{14} & \alpha^{13} & \dots & \alpha^{1} & \alpha^{0} \end{pmatrix} =$$

$$= \begin{pmatrix} \alpha^{10} & \alpha^{5} & \alpha^{0} & \alpha^{10} & \alpha^{5} & \alpha^{0} & \alpha^{10} & \alpha^{5} & \alpha^{0} & \alpha^{10} & \alpha^{5} & \alpha^{0} & \alpha^{10} & \alpha^{5} & \alpha^{0} \\ \alpha^{12} & \alpha^{9} & \alpha^{6} & \alpha^{3} & \alpha^{0} & \alpha^{12} & \alpha^{9} & \alpha^{6} & \alpha^{3} & \alpha^{0} & \alpha^{12} & \alpha^{9} & \alpha^{6} & \alpha^{3} & \alpha^{0} \\ \alpha^{14} & \alpha^{13} & \alpha^{12} & \alpha^{11} & \alpha^{10} & \alpha^{9} & \alpha^{8} & \alpha^{7} & \alpha^{6} & \alpha^{5} & \alpha^{4} & \alpha^{3} & \alpha^{2} & \alpha^{1} & \alpha^{0} \end{pmatrix}. \quad (3.153)$$

Із співвідношення (3.153) видно, що для примітивного елементу α^3

періодичність циклотомічного класу складає 5, а для примітивного елементу α^3, відповідно, 3. Відповідність між синдромами помилок та станом помилки для коду БЧХ (15, 5) показана у таблиці 3.10 [52, 53].

Таблиця 3.10 – Відповідність між синдромами та станом помилки для коду БЧХ (15, 7)

Синдром			Стан помилки
S_2	S_1	S_0	
0	0	0	Помилка відсутня
$\left(\alpha^{14}\right)^5 = \alpha^{10}$	$\left(\alpha^{14}\right)^3 = \alpha^{12}$	α^{14}	Помилка у першому біті
$\left(\alpha^{13}\right)^5 = \alpha^{5}$	$\left(\alpha^{13}\right)^3 = \alpha^{9}$	α^{13}	Помилка у другому біті
$\left(\alpha^{12}\right)^5 = \alpha^{0}$	$\left(\alpha^{12}\right)^3 = \alpha^{6}$	α^{12}	Помилка у третьому біті
$\left(\alpha^{11}\right)^5 = \alpha^{10}$	$\left(\alpha^{11}\right)^3 = \alpha^{3}$	α^{11}	Помилка у четвертому біті
$\left(\alpha^{10}\right)^5 = \alpha^{5}$	$\left(\alpha^{10}\right)^3 = \alpha^{0}$	α^{10}	Помилка у п’ятому біті
$\left(\alpha^{9}\right)^5 = \alpha^{0}$	$\left(\alpha^{9}\right)^3 = \alpha^{12}$	α^{9}	Помилка у шостому біті
$\left(\alpha^{8}\right)^5 = \alpha^{10}$	$\left(\alpha^{8}\right)^3 = \alpha^{9}$	α^{8}	Помилка у сьомому біті
$\left(\alpha^{7}\right)^5 = \alpha^{5}$	$\left(\alpha^{7}\right)^3 = \alpha^{6}$	α^{7}	Помилка у восьмому біті
$\left(\alpha^{6}\right)^5 = \alpha^{0}$	$\left(\alpha^{6}\right)^3 = \alpha^{3}$	α^{6}	Помилка у дев’ятому біті
$\left(\alpha^{5}\right)^5 = \alpha^{10}$	$\left(\alpha^{5}\right)^3 = \alpha^{0}$	α^{5}	Помилка у десятому біті
$\left(\alpha^{4}\right)^5 = \alpha^{5}$	$\left(\alpha^{4}\right)^3 = \alpha^{12}$	α^{4}	Помилка в одинадцятому біті
$\left(\alpha^{3}\right)^5 = \alpha^{0}$	$\left(\alpha^{3}\right)^3 = \alpha^{9}$	α^{3}	Помилка у дванадцятому біті
$\left(\alpha^{2}\right)^5 = \alpha^{10}$	$\left(\alpha^{2}\right)^3 = \alpha^{6}$	α^{2}	Помилка у тринадцятому біті
$\left(\alpha^{1}\right)^5 = \alpha^{5}$	$\left(\alpha^{1}\right)^3 = \alpha^{3}$	α^{1}	Помилка у чотирнадцятому біті
$\left(\alpha^{0}\right)^5 = \alpha^{0}$	$\left(\alpha^{0}\right)^3 = \alpha^{0}$	α^{0}	Помилка у п’ятнадцятому біті

Згідно із співвідношенням (3.153) біти синдрому помилки мають

наступний вигляд [52, 53]:

$$S_2 = \left(\alpha^{14}\right)^5 \cdot e_{14} + \left(\alpha^{13}\right)^5 \cdot e_{13} + \ldots + \left(\alpha^{1}\right)^5 \cdot e_{1} + \left(\alpha^{0}\right)^5 \cdot e_{0}. \qquad (3.154)$$

$$S_1 = \left(\alpha^{14}\right)^3 \cdot e_{14} + \left(\alpha^{13}\right)^3 \cdot e_{13} + \ldots + \left(\alpha^{1}\right)^3 \cdot e_{1} + \left(\alpha^{0}\right)^3 \cdot e_{0}. \qquad (3.155)$$

$$S_0 = \alpha^{14} \cdot e_{14} + \alpha^{13} \cdot e_{13} + \ldots + \alpha^{1} \cdot e_{1} + \alpha^{0} \cdot e_{0}. \qquad (3.156)$$

Якщо позначити елементи групи $GF(2^4)$, що відображують положення трьох помилкових бітів як α^i, α^j та α^k, тоді рівняння, яке визначає розташування помилок, записується наступним чином [52, 53]:

$$S(x) = \left(x - \alpha^i\right) \cdot \left(x - \alpha^j\right) \cdot \left(x - \alpha^k\right) = x^3 + A \cdot x^2 + B \cdot x + C. \qquad (3.157)$$

Тепер для пошуку помилки необхідно знайти коефіцієнти A, B та C.

Розглянемо приклад пошуку потрійної помилки у коді БЧХ (15, 5) [52, 53].

Приклад 3.19. Знайти помилки у коді БЧХ (15, 5), якщо вони виникли у п’ятому, десятому та дванадцятому бітах.

Згідно із умовою задачі, вектор помилки **E** має вигляд $\mathbf{E} = (0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,0\,1\,0\,0\,0)$. Маючи на увазі, що елементи вектора **E** нумеруються зліва, а перший елемент є нульовим, зрозуміло, що $e_{10} = 1$, $e_5 = 1$ та $e_3 = 1$, а решта бітів вектора помилки дорівнюють нулю. Тоді, згідно із співвідношеннями (3.154) – (3.156), біти синдрому помилки мають вигляд:

$$S_2 = \left(\alpha^{10}\right)^5 + \left(\alpha^{5}\right)^5 + \left(\alpha^{3}\right)^5; \qquad (3.158)$$

$$S_1 = \left(\alpha^{10}\right)^3 + \left(\alpha^{5}\right)^3 + \left(\alpha^{3}\right)^3; \qquad (3.159)$$

$$S_0 = \alpha^{10} + \alpha^{5} + \alpha^{3}. \qquad (3.160)$$

Звернемося до даних, наведених у таблиці 3.10. Зрозуміло, що помилці у п’ятому біту відповідає елемент групи Галуа $GF(2^4)$ α^{10}, десятому – елемент α^5, а дванадцятому – α^3. Отже, прирівнюючи для цих елементів суму (3.157) до нуля, отримуємо [52, 53]:

$$S\left(\alpha^{10}\right) = \left(\alpha^{10}\right)^3 + A \cdot \left(\alpha^{10}\right)^2 + B \cdot \left(\alpha^{10}\right) + C = 0. \qquad (3.161)$$

$$S\left(\alpha^5\right)=\left(\alpha^5\right)^3+A\cdot\left(\alpha^5\right)^2+B\cdot\left(\alpha^5\right)+C=0. \quad (3.162)$$

$$S\left(\alpha^3\right)=\left(\alpha^3\right)^3+A\cdot\left(\alpha^3\right)^2+B\cdot\left(\alpha^3\right)+C=0. \quad (3.163)$$

Із отриманих співвідношень (3.158) – (3.163) можна знайти невідомі коефіцієнти *A*, *B* та *C*. Оскільки, згідно із рівняннями (3.161) – (3.163),

$$S\left(\alpha^{10}\right)+S\left(\alpha^5\right)+S\left(\alpha^3\right)=0,$$

враховуючи співвідношення (3.157), можна записати наступну тотожність для степенів примітивного елемента α:

$$\left\{\left(\alpha^{10}\right)^3+\left(\alpha^5\right)^3+\left(\alpha^3\right)^3\right\}+A\cdot\left\{\left(\alpha^{10}\right)^2+\left(\alpha^5\right)^2+\left(\alpha^3\right)^2\right\}+$$

$$+B\cdot\left\{\left(\alpha^{10}\right)+\left(\alpha^5\right)+\left(\alpha^3\right)\right\}+C=0. \quad (3.164)$$

Після піднесення до квадрату обох частин рівності (3.160) отримуємо:

$$S_0^2=\left(\alpha^{10}+\alpha^5+\alpha^3\right)^2,$$

звідки, враховуючи рівності (3.159) та (3.160), маємо:

$$S_1+A\cdot S_0^2+B\cdot S_0+C=0. \quad (3.165)$$

Із співвідношень (3.158) – (3.160) можна записати:

$$\alpha^{10}\cdot S\left(\alpha^{10}\right)+\alpha^5\cdot S\left(\alpha^5\right)+\alpha^3\cdot S\left(\alpha^3\right)=0, \quad (3.166)$$

звідки, враховуючи (3.161) – (3.163):

$$\left\{\left(\alpha^{10}\right)^4+\left(\alpha^5\right)^4+\left(\alpha^3\right)^4\right\}+A\cdot\left\{\left(\alpha^{10}\right)^3+\left(\alpha^5\right)^3+\left(\alpha^3\right)^3\right\}+$$

$$+B\cdot\left\{\left(\alpha^{10}\right)^2+\left(\alpha^5\right)^2+\left(\alpha^3\right)^2\right\}+C\cdot\left\{\left(\alpha^{10}\right)+\left(\alpha^5\right)+\left(\alpha^3\right)\right\}=0. \quad (3.167)$$

Після алгебраїчних перетворень з використанням співвідношень (3.159), (3.160) та (3.163), враховуючи тотожність

$$S_0^4=\left(\left(\alpha^{10}\right)^2+\left(\alpha^5\right)^2+\left(\alpha^3\right)^2\right)^2=\left(\left(\alpha^{10}\right)+\left(\alpha^5\right)+\left(\alpha^3\right)\right)^4, \quad (3.168)$$

отримуємо

$$S_0^4 + S_1 \cdot A + S_0^2 \cdot B + S_0 \cdot C = 0. \quad (3.169)$$

Враховуючи співвідношення (3.158) – (3.160), можна записати [52, 53]:

$$\left(\alpha^{10}\right)^2 \cdot S\left(\alpha^{10}\right) + \left(\alpha^5\right)^2 \cdot S\left(\alpha^5\right) + \left(\alpha^3\right)^2 \cdot S\left(\alpha^3\right) = 0, \quad (3.170)$$

звідки:

$$\left\{\left(\alpha^{10}\right)^5 + \left(\alpha^5\right)^5 + \left(\alpha^3\right)^5\right\} + A \cdot \left\{\left(\alpha^{10}\right)^4 + \left(\alpha^5\right)^4 + \left(\alpha^3\right)^4\right\} +$$

$$+ B \cdot \left\{\left(\alpha^{10}\right)^3 + \left(\alpha^5\right)^3 + \left(\alpha^3\right)^3\right\} + C \cdot \left\{\left(\alpha^{10}\right)^2 + \left(\alpha^5\right)^2 + \left(\alpha^3\right)^2\right\} = 0. \quad (3.171)$$

Після алгебраїчних перетворень отриманого співвідношення (3.171) з урахуванням співвідношень (3.159), (3.160), (3.165), маємо [52, 53]:

$$S_2 + A \cdot S_0^4 + B \cdot S_1 + S_0^2 \cdot C = 0. \quad (3.172)$$

Об’єднаємо рівняння з бітами синдромів помилок (3.165), (3.169) та (3.172) до однієї системи рівнянь [52, 53]:

$$\begin{cases} A \cdot S_0^2 + B \cdot S_0 + C = S_1; \\ S_1 \cdot A + S_0^2 \cdot B + S_0 \cdot C = S_0^4; \\ A \cdot S_0^4 + B \cdot S_1 + S_0^2 \cdot C = S_2. \end{cases} \quad (3.173)$$

За умови відомих бітів синдрому S_2, S_1 та S_0, можна, розв’язавши систему рівнянь (3.173), знайти значення коефіцієнтів A, B та C, тобто, положення помилкових бітів. Запишемо алгебраїчні вирази для бітів синдрому помилок, які випливають із співвідношень (3.158) – (3.160) [52, 53]:

$$S_2 = \left(\alpha^{10}\right)^5 + \left(\alpha^5\right)^5 + \left(\alpha^3\right)^5 = \alpha^{50} + \alpha^{25} + \alpha^{15} = \alpha^5 + \alpha^{10} + 1 =$$

$$= \left(\alpha^2 + \alpha\right) \cdot \left(\alpha^2 + \alpha + 1\right) = 2\alpha^2 + 2\alpha + 2 = 0. \quad (3.174)$$

$$S_1 = \left(\alpha^{10}\right)^3 + \left(\alpha^5\right)^3 + \left(\alpha^3\right)^3 = \alpha^{30} + \alpha^{15} + \alpha^9 = \alpha^9 + 1 + 1 = \alpha^9. \quad (3.175)$$

$$S_0 = \alpha^{10} + \alpha^5 + \alpha^3 = \left(\alpha^2 + \alpha + 1\right) + \left(\alpha^2 + \alpha\right) =$$

$$\alpha^3 + 2\alpha^2 + 2\alpha + 1 = \alpha^3 + 1 = \alpha^{14}. \quad (3.176)$$

Відповідно, значення S_0^2 та S_0^4, які входять до системи рівнянь (3.173), можна обчислити наступним чином [52, 53]:

$$S_0^2 = \left(\alpha^{14}\right)^2 = \alpha^{28} = \alpha^{28-15} = \alpha^{13};$$

$$S_0^4 = \left(\alpha^{13}\right)^2 = \alpha^{26} = \alpha^{26-15} = \alpha^{11}. \quad (3.177)$$

Тепер перепишемо систему рівнянь (3.173) з урахуванням співвідношень (3.174) – (3.177) [52, 53]:

$$A \cdot \alpha^{13} + B \cdot \alpha^{14} + C = \alpha^9, \quad (3.178)$$

$$A \cdot \alpha^9 + B \cdot \alpha^{13} + C \cdot \alpha^{14} = \alpha^{11}, \quad (3.179)$$

$$A \cdot \alpha^{11} + B \cdot \alpha^9 + C \cdot \alpha^{13} = 0. \quad (3.180)$$

Із записаного рівняння (3.180) випливає, що:

$$B = \frac{A \cdot \alpha^{11} + C \cdot \alpha^{13}}{\alpha^9} = A \cdot \alpha^2 + C \cdot \alpha^4. \quad (3.181)$$

Підставимо вираз для B (3.181) у співвідношення (3.178):

$$A \cdot \alpha^{13} + \left(A \cdot \alpha^2 + C \cdot \alpha^4\right) \cdot \alpha^{14} + C = A \cdot \left(\alpha^{13} + \alpha^{16}\right) + C \cdot \left(\alpha^{18} + 1\right) = \alpha^9. \quad (3.182)$$

Враховуючи, що

$$\left(\alpha^{13} + \alpha^{16}\right) = \left(\alpha^3 + \alpha^2 + 1\right) + \alpha = \alpha^{12}, \qquad \left(\alpha^{18} + 1\right) = \left(\alpha^3 + 1\right) = \alpha^{14}, \quad (3.183)$$

перепишемо співвідношення (3.182), з урахуванням (3.183), у вигляді:

$$A \cdot \alpha^{12} + C \cdot \alpha^{14} = \alpha^9. \quad (3.184)$$

Поділивши обидві частини співвідношення (3.184) на α^9, отримуємо:

$$A \cdot \alpha^3 + C \cdot \alpha^5 = 1. \quad (3.185)$$

Після підстановки виразу для B (3.181) в формулу (3.179) та відповідних алгебраїчних перетворень, отримуємо наступний результат:

$$A \cdot \alpha^9 + \left(A \cdot \alpha^2 + C \cdot \alpha^4\right) \cdot \alpha^{13} + C \cdot \alpha^{14} =$$

$$= A \cdot \left(\alpha^9 + \alpha^{15}\right) + C \cdot \left(\alpha^{17} + \alpha^{14}\right) = \alpha^{11}. \quad (3.186)$$

Оскільки

$$\left(\alpha^9 + \alpha^{15}\right) = \left(\alpha^3 + \alpha\right) + 1 = \alpha^7, \quad (3.187)$$

$$\left(\alpha^{17} + \alpha^{14}\right) = \alpha^2 + \left(\alpha^3 + 1\right) = \alpha^{18}, \quad (3.188)$$

з урахуванням рівностей (3.187), (3.188), співвідношення (3.186) можна переписати у вигляді:

$$A = \alpha^9 \cdot C + \alpha^4. \quad (3.189)$$

Підставимо отримане значення змінної A, яке задане співвідношенням (3.189), до співвідношення (3.185). Відповідно маємо:

$$\left(\alpha^9 \cdot C + \alpha^4\right) \cdot \alpha^3 + C \cdot \alpha^5 = 1,$$

або

$$\left(\alpha^9 + \alpha^5\right) \cdot C = \alpha^7 + 1. \tag{3.190}$$

Враховуючи, що

$$\left(\alpha^9 + \alpha^5\right) = \left(\alpha^3 + \alpha\right) + \left(\alpha^2 + \alpha\right) = \alpha^6, \tag{3.191}$$

$$\left(\alpha^7 + 1\right) = \left(\alpha^3 + \alpha + 1\right) + 1 = \alpha^9. \tag{3.192}$$

Для коефіцієнту C остаточно маємо:

$$\alpha^6 \cdot C = \alpha^9; \Rightarrow C = \alpha^3. \tag{3.193}$$

Тоді із співвідношень (3.181), (3.189), (3.193), отримуємо значення коефіцієнтів A та B:

$$A = \alpha^9 \cdot C + \alpha^4 = \alpha^9 \cdot \alpha^3 + \alpha^4 = \alpha^9 + \alpha^4 = \left(\alpha^3 + \alpha\right) + \left(\alpha + 1\right) = \alpha^3 + 1 = \alpha^{14}.$$

$$B = \alpha^2 \cdot \alpha^{14} + \alpha^4 \cdot \alpha^3 = \alpha^{16} + \alpha^7 = \alpha + \left(\alpha^3 + \alpha + 1\right) = \alpha^3 + 1 = \alpha^{14}.$$

Тобто, рівняння розташування помилок у коді БЧХ (15, 5) має вигляд:

$$S(x) = x^3 + \alpha^{14} \cdot x^2 + \alpha^{14} \cdot x + \alpha^3. \tag{3.194}$$

Корені отриманого рівняння розташування помилок (3.194) можна знайти методом підбору. Підставляючи до цього рівняння елементи групи Галуа $GF(2^4)$, отримуємо [52, 53]:

$$S\left(\alpha^{10}\right) = \left(\alpha^{10}\right)^3 + \alpha^{14} \cdot \left(\alpha^{10}\right)^2 + \alpha^{14} \cdot \alpha^{10} + \alpha^3 = \alpha^{30} + \alpha^{34} + \alpha^{24} + \alpha^3 =$$

$$= 1 + \alpha^4 + \alpha^9 + \alpha^3 = 0. \tag{3.195}$$

$$S\left(\alpha^5\right) = \left(\alpha^5\right)^3 + \alpha^{14} \cdot \left(\alpha^5\right)^2 + \alpha^{14} \cdot \alpha^5 + \alpha^3 = \alpha^{15} + \alpha^{24} + \alpha^{19} + \alpha^3 =$$

$$= 1 + \alpha^4 + \alpha^9 + \alpha^3 = 0. \tag{3.196}$$

$$S\left(\alpha^3\right) = \left(\alpha^3\right)^3 + \alpha^{14} \cdot \left(\alpha^3\right)^2 + \alpha^{14} \cdot \alpha^3 + \alpha^3 = \alpha^9 + \alpha^{20} + \alpha^{17} + \alpha^3 =$$

$$= \alpha^9 + \alpha^5 + \alpha^2 + \alpha^3 = \left(\alpha^3 + \alpha\right) + \left(\alpha^2 + \alpha\right) + \alpha^2 + \alpha^3 = 0. \tag{3.197}$$

Із співвідношень (3.195) – (3.197) зрозуміло, що значення α^{10}, α^5 та α^3 є коренями рівняння розташування помилок (3.194). Це означає, що помилки виникли у п’ятому, десятому та дванадцятому бітах.

Описаний вище алгоритм декодування коду БЧХ (15, 5), блок-схема якого наведена на рис. 3.21. Зрозуміло, що цей алгоритм дуже схожий на алгоритм декодування кодів БЧХ (15, 5), наведений на рис. 3.18. Схема ділення

кодової комбінації на породжувальний поліном наведена на рис. 3.22, а принципова електрична схема декодера коду БЧХ (15, 5) – на рис. 3.23 [52, 53].

Рис. 3.21 Алгоритм декодування коду БЧХ (15, 5)

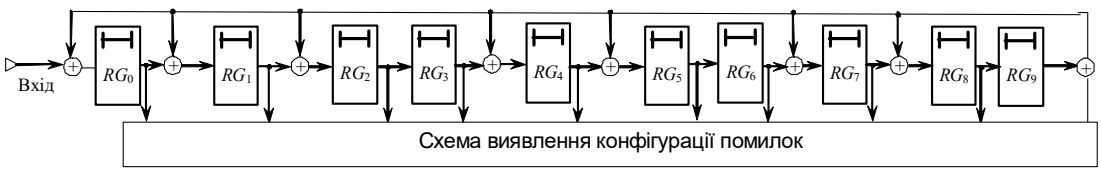


Рис. 3.22 Схема ділення кодової комбінації на породжувальний поліном

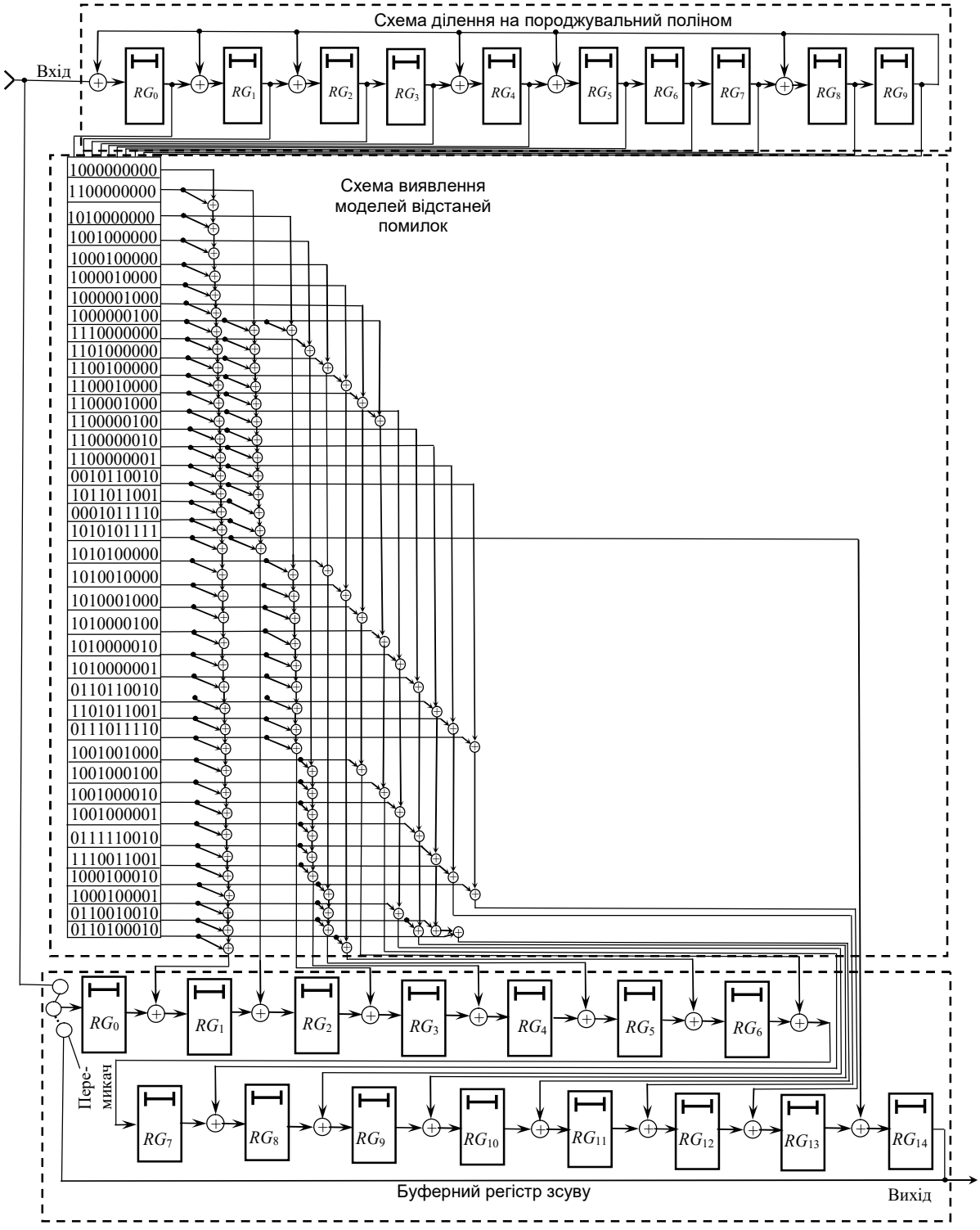


Рис. 3.23 Принципова електрична схема декодера для коду БЧХ (15, 5)

Слід відзначити, що код БЧХ (15, 5) із виправленням потрійної помилки, згідно із співвідношеннями (3.21) має більший коефіцієнт надлишковості R_n = 0,8, для коду із виправленням подвійної помилки (15, 7) $R_n \approx 0{,}7$. Проте, оскільки

коректувакльна здатність коду (15, 5) є вищою, його використання у деякій мірі дозволяє зменшити завантаженість зашумленого каналу зв'язку, де існує велика імовірність бітової помилки. Комп'ютерні засоби для формування кодів БЧХ (15, 7) та (15, 5) та декодування їхніх послідовностей розглядатимуться у наступному підрозділі.

3.3.7 Комп'ютерна реалізація алгоритмів формування кодів Боуза – Чоудхурі – Хоквінгема та декодування їхніх кодових послідовностей

Ітераційні алгоритми декодування кодів БЧХ, які були розглянуті у підрозділі 3.3.4, для апаратної реалізації в електронній апаратурі є дійсно досить ефективними, проте їхня програмна реалізація ускладнюється тим, що алгоритми Форні та Берлекемпа – Мессі передбачають комбінаторне перебирання кодових комбінацій. Як відмічалось у другій частині посібника, такі алгоритми з обчислювальної точки зору зазвичай є досить ресурсоємними. Проте, як було зазначено у підрозділі 3.3.2, для декодування кодів БЧХ також можуть бути використані всі алгоритми декодування циклічних кодів, які розглядалися у підрозділі 2.5. Наприклад, за умови відомої кількості помилок, які виправляються, для декодування систематичних кодів БЧХ можна застосовувати алгоритм ділення кодової комбінації на твірний поліном із аналізом ваги остач та із послідовним циклічним зсувом розрядів ліворуч. Цей алгоритм був розглянутий у підрозділі 2.5.1 та наведений на рис. 2.16.

Скористаємося цим алгоритмом декодування для написання програми, призначеної для формування та декодування розглянутих вище кодів БЧХ (15, 7) та (15, 5). Обрання саме цього алгоритму також обумовлено тим, що процедура ділення двійкових комбінацій стовпчиком вже була реалізована та перевірена для декодування циклічних кодів, відповідний код програми **`Bin_Division`** наведений у додатку К. Особливості комп'ютерної реалізації цієї програми були описані у підрозділі 2.5.8.

Спочатку наведемо приклади використання для декодування кодів БЧХ алгоритму ділення за модулем два з визначенням ваги остач та послідовним циклічним зсувом кодової комбінації [80].

Приклад 3.20. Знайти помилки у кодовій комбінації 111001100000100 коду БЧХ (15, 7), якщо вони існують, та визначити, яка двійкова послідовність була закодована.

Зрозуміло, що прийнятій кодовій комбінації 111001100000100 відповідає поліном $x^{14} + x^{13} + x^{12} + x^9 + x^8 + x^2$. Поділимо цей поліном на твірний поліном коду БЧХ (15, 7), $x^8 + x^7 + x^6 + x^4 + 1$, заданий співвідношенням (3.132). У двійковій формі такий поліном записується як послідовність бітів 111010001. Процес ділення цих двійкових послідовностей показаний на рис. 3.24. Зрозуміло, що у прийнятій кодовій комбінації помилок немає, оскільки остача від ділення дорівнює нулю, а закодованому числу відповідає послідовність 1110011, або поліном $x^6 + x^5 + x^4 + x + 1$. Цілком зрозуміло, що вхідна послідовність для систематичних кодів БЧХ – це останні k бітів сформованого коду.

$$
\begin{array}{r|l}
111001100000100 & 111010001 \\
\oplus\ 111010001 & \hline
\cline{1-1} 111010001 & 1000100 \\
\oplus\ 111010001 & \\
\cline{1-1} 0 &
\end{array}
$$

Рис. 3.24 Наочна ілюстрація процесу ділення правильної кодової комбінації коду БЧХ (15, 7) на твірний поліном для прикладу 3.20

Приклад 3.21. Знайти помилки у кодовій комбінації 111001100010100 коду БЧХ (15, 7), якщо вони існують, та визначити, яка двійкова послідовність була закодована.

Для даного прикладу процес пошуку помилки повністю ідентичний процесу, описаному у прикладі 2.30. Необхідно стовпчиком поділити прийняту кодову комбінацію 111001100010100 на твірний поліном, який у числовому вигляді записується як 111010001. Відповідний процес ділення показаний на рис. 3.25.

$$
\begin{array}{r|l}
111001100010100 & 111010001 \\
\oplus\ 111010001 & \hline
\cline{1-1} 111010101 & 10001 \\
\oplus\ 111010001 & \\
\cline{1-1} 10000 &
\end{array}
$$

Рис. 3.25 Наочна ілюстрація процесу ділення спотвореної кодової комбінації коду БЧХ (15, 7) на твірний поліном для прикладу 3.21

Як видно з результату ділення, у цьому разі остача існує та складає 10000. Оскільки вага остачі складає 1, а кількість помилок, які виправляє код БЧХ (15, 7), становить $S = 2$, процес пошуку помилки можна вважати завершеним. Аналіз отриманої кодової послідовності через її ділення на твірний поліном показав, що вона містить одиночну помилку у п’ятому

розряді. Процес виправлення помилки полягає у тому, що необхідно додати за модулем 2 отриману остачу до спотвореної прийнятої кодової комбінації. Відповідно маємо:

$$111001100010100 \oplus 10000 = 111001100000100.$$

Згідно із результатом, отриманим у прикладі 3.20, зрозуміло, що ця кодова комбінація відповідає початковій вхідній послідовності 1110011.

Приклад 3.22. Знайти помилки у кодовій комбінації 111001000001100 коду БЧХ (15, 7), якщо вони існують, та визначити, яка двійкова послідовність була закодована.

У цьому випадку процес визначення та виправлення помилки є більш складним та ітераційним. Розглянемо його послідовно. Перший результат ділення прийнятої кодової комбінації на твірний поліном показаний на рис. 3.26.

$$\begin{array}{r|l} 111001000001100 & 111010001 \\ \oplus\ 111010001 & \hline 1000101 \\ \hline 110010011 & \\ \oplus\ 111010001 & \\ \hline 100001000 & \\ \oplus\ 111010001 & \\ \hline 11011001 = R_1 & \end{array}$$

Рис. 3.26 Наочна ілюстрація процесу ділення спотвореної кодової комбінації коду БЧХ (15, 7) на твірний поліном для прикладу 3.22

Із наведеного результату видно, що вага остачі від ділення $w(R_1) = w(11011001) = 5 > 2$. Це означає, що необхідно виконувати циклічний зсув отриманої кодової комбінації на одну позицію ліворуч та повторювати процес ділення, доки не буде виконана умова $w(R_n) \le 2$. Після цього слід додати за модулем 2 отриману остачу R_n до зсунутої кодової комбінації та здійснити зворотній зсув праворуч. Відповідний ітераційний процес виправлення помилки у циклічному коді був розглянутий у підрозділі 2.5.3 та показаний на рис. 2.17 – 2.24. Як було відмічено у підрозділі 2.5.3, суттєвою особливістю цього ітераційного процесу є те, що в ході циклічного зсуву бітів кодової послідовності ліворуч не можна ігнорувати нулі, які стоять у старших розрядах. Це пов’язано з тим, що на наступних ітераціях ці нулі переходять на початок кодової комбінації та суттєво впливають на остаточний результат ділення. Відповідний ітераційний процес ділення з циклічним зсувом кодової

комбінації ліворуч для прикладу, який розглядається, наочно показаний на рис. 3.27, а – л.

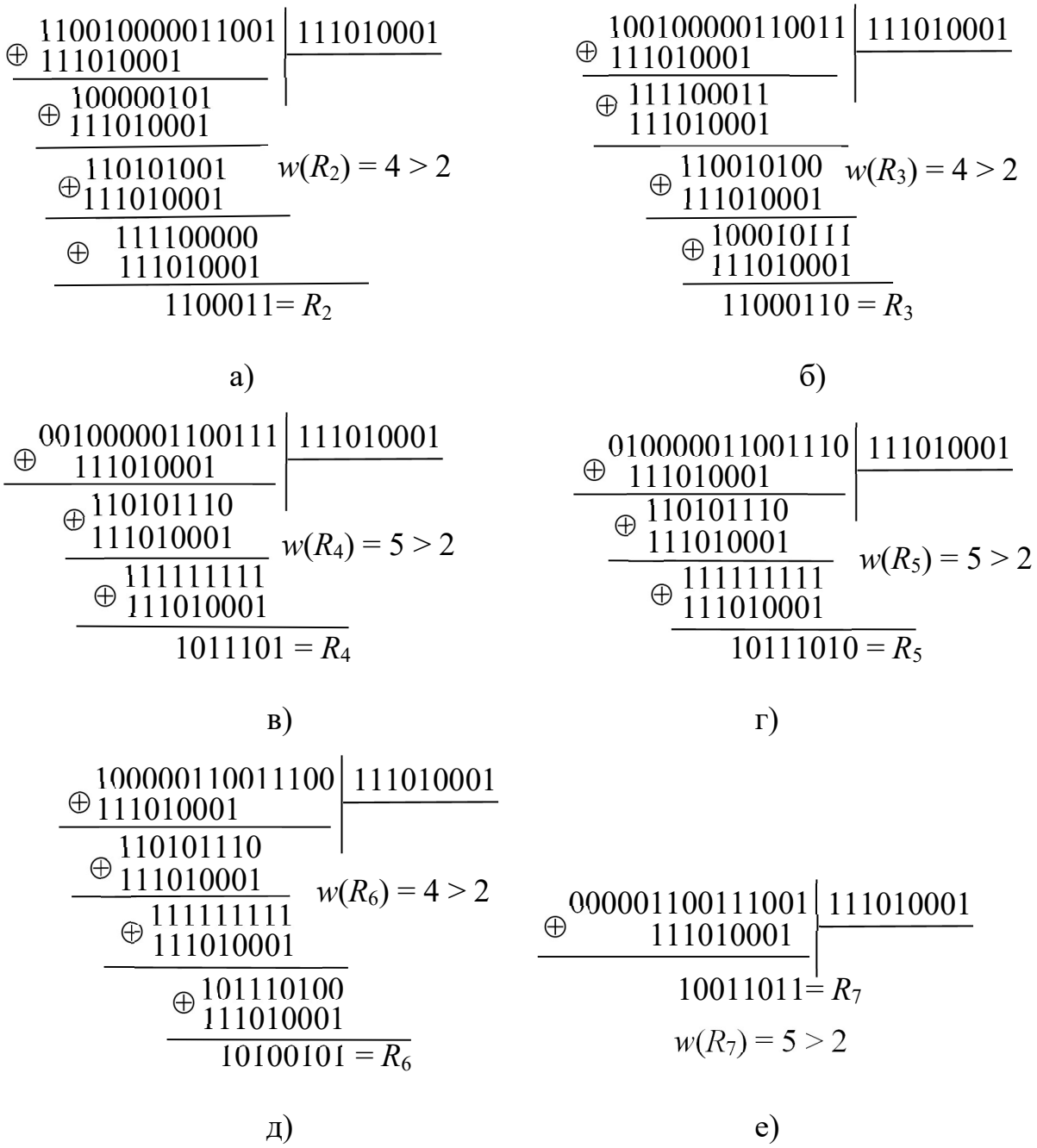


Рис. 3.27 Наочна ілюстрація ітераційного процесу ділення спотвореної кодової комбінації коду БЧХ (15, 7) на твірний поліном із послідовним зсувом бітів ліворуч

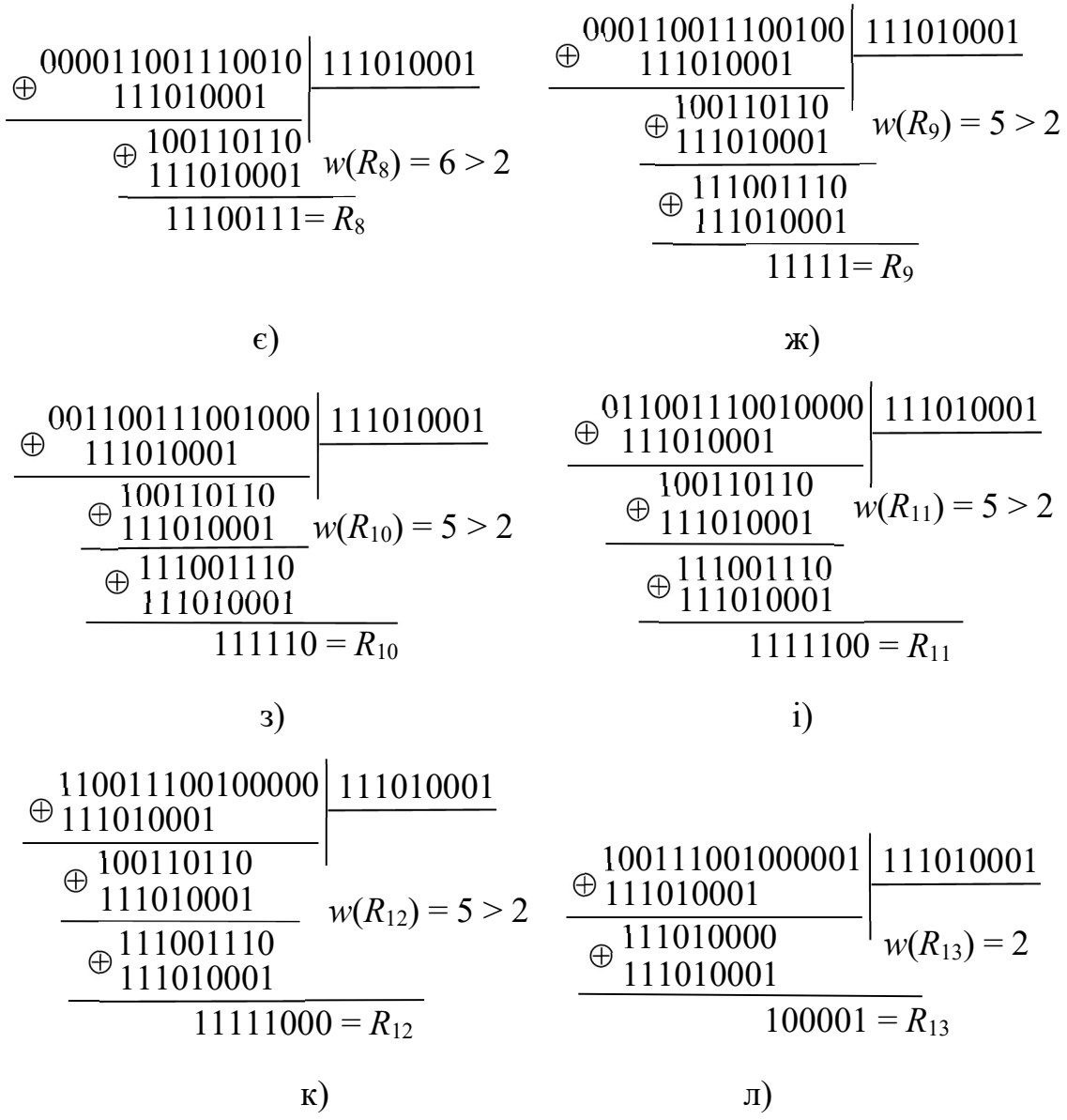


Рис. 3.27 Наочна ілюстрація ітераційного процесу ділення спотвореної кодової комбінації коду БЧХ (15, 7) на твірний поліном із послідовним зсувом бітів ліворуч (продовження)

Тобто, після дванадцяти циклічних зсувів спотвореної кодової комбінації ліворуч отримуємо в результаті ділення на твірний поліном 111010001 остачу R_{13}=100001, яка має вагу $w(R_{13}) = 2$.

Тепер, згідно із алгоритмом, блок-схема якого наведена на рис. 2.16, необхідно додати за модулем два отриману остачу до зсунутої кодової комбінації та зробити її зворотний циклічний зсув на 12 позицій праворуч, або,

для п’ятнадцятирозрядної двійкової комбінації коду БЧХ (15, 7), на 3 позиції ліворуч. Відповідно, маємо:

100111001000001 ⊕ 100001 = 100111001100000

Після зсуву виправленої кодової комбінації 100111001100000 на 3 позиції ліворуч отримуємо остаточний п’ятнадцятирозрядний код БЧХ (15, 7) 111001100000100, який не містить помилок. Оскільки ця ж сама кодова комбінація розглядалась у прикладі 3.20, зрозуміло, що вона є кодом двійкового числа 1110011.

Для реалізації алгоритмів формування кодів БЧХ та декодування їхніх послідовностей у програмі **BCH**, яка наведена у додатку М, використані наступні змінні.

1. Вхідні параметри програми.

vin – вектор вхідної кодової послідовності.

nop – тип операції. Можливі значення цього параметру: 1 – формування коду БЧХ, 2 – декодування кодової послідовності.

TOC – тип коду. Цей параметр може мати такі значення: 1 – для коду БЧХ (15, 7) та 2 – для коду БЧХ (15, 5).

2. Вхідні параметри програми.

vout – результати виконання програми, тобто, комбінація коду БЧХ за умови **nop=1** або закодоване інформаційне слово за умови **nop=2**.

3. Проміжні змінні.

nv – довжина вхідної послідовності.

TP – вектор, який відповідає твірному поліному.

RES1M – результат ділення кодового слова на незвідний твірний поліном. Результат записується у вигляді матриці з двома рядками, у першому з яких розташована частка від ділення, а у другому – остача. Для формування матриці ділення використовується функція **Bin_Division**.

VS – шаблон з дописаними правими та лівими нулями, який створюється на основі вхідного вектора та використовується для формування коду БЧХ.

MC – результат ділення вектора **VS** на твірний поліном **TP**, який формується з використанням функції **Bin_Division**.

VinShift – вектор, який формується із вхідного вектора **vin** в результаті циклічного зсуву кодової комбінації.

Rem – остача від ділення кодової комбінації на твірний поліном.

SRem – вага остачі **Rem**.

Pos – вектор позицій помилок, знайдених у кодовій комбінації.

ii, iii, jjj – допоміжні змінні.

В результаті роботи програми формується вихідний вектор, який залежить від виконуваної операції. Якщо виконувалась операція кодування, вихідним вектором є сформований код БЧХ, а у разі виконання операція декодування – закодована кодова послідовність. Така особливість роботи написаної програми дозволяє використовувати її у складних програмних комплексах, призначених для моделювання бігітофункціональних цифрових електронних систем та для аналізу особливостей їхньої роботи. У разі наявності помилки у коді БЧХ наприкінці роботи програми на екран виводиться не лише закодована послідовність, але й також повідомлення про помилку та номер помилкового розряду.

Алгоритм обчислення номера помилкового розряду у програмі **BCH** має певні особливості. За описаним алгоритмом помилки виправляються у зсунутій кодовій комбінації через додавання отриманої остачі, а після цього розряди виправленої кодової комбінації пересуваються у зворотному напрямку.

По-перш за все, необхідно домовитись про порядок нумерації розрядів. Оскільки у всіх вхідних та вихідних кодових послідовностях розряди нумеруються у прямому порядку, зправа-наліво, будемо дотримуватись такого ж порядку нумерації. Але оскільки номера елементів векторів нумеруються у зворотному порядку, зліва-направо, необхідно виводити не порядковий номер помилкового елемента вектора $n_{пв}$, а відповідний номер розряду кодової комбінації $n_{пр}$, який розраховується як $n_{пр} = 15 - n_{пв} + 1$.

Іншою проблемою є визначення розташування помилкових розрядів у початковій та остаточній кодовій комбінації за умови відомого їхнього положення у зсунутій комбінації. Річ у тому, що розряди кодової комбінації в процесі ділення циклічно зсуваються ліворуч, тому на відповідній ітерації п'ятнадцятий розряд переходить на першу позицію. Припустимо, що у коді БЧХ (15, 7) помилка виявлена на першій та третій позиції зсунутої кодової комбінації, а кількість позицій зсуву становить $n_{\text{зс}} = 2$. Тоді для визначення положення цих розрядів у коді БЧХ необхідно відняти від поточних значень кількість позицій зсуву, але якщо для значення зсунутого помилкового розряду $n_{\text{пр}}^{\text{зс}} = 3$ відповідний результат буде $n_{\text{пр}} = n_{\text{пр}}^{\text{зс}} - 2 = 3 - 2 = 1$, то за умови $n_{\text{пр}}^{\text{зс}} = 1$ маємо $n_{\text{пр}} = n_{\text{пр}}^{\text{зс}} - 2 = 1 - 2 = -1$. Це означає, що насправді помилковий розряд знаходиться на 14 позиції, і для отримання цього значення необхідно додати від'ємну величину $n_{\text{пр}}$ до числа 15, тобто, до загальної кількості розрядів. Узагальнену формулу для отримання номеру помилкового розряду можна записати наступним чином:

$$n_{\text{пр}} = \begin{cases} n_{\text{пр}}^{\text{зс}} - n_{\text{зс}}, & n_{\text{пр}}^{\text{зс}} > n_{\text{зс}}; \\ 15 - n_{\text{пр}}^{\text{зс}} - n_{\text{зс}}, & n_{\text{пр}}^{\text{зс}} \le n_{\text{зс}}. \end{cases} \tag{3.198}$$

Записане співвідношення (3.198) цілком відповідає теорії пересувань та теорії циклотомічних класів, які розглядалися у другому та третьому розділах другої частини посібника [48]. Зрозуміло, що для операції циклічного пересування завжди виконується умова $n_{\text{зс}} < n$, де n – загальна кількість розрядів у комбінації коду БЧХ.

Алгоритм роботи програми, наведеної у додатку М, представлений на рис. 3.28. Як було відмічено вище, в основу цього алгоритму покладена процедура ділення отриманої кодової послідовності на твірний поліном із визначенням ваги остатків та циклічним зсувом, яка була описана у підрозділі 3.3.2. Для виконання алгебраїчної операції ділення двох поліномів за модулем 2 використана функція **`Bin_Division`**, яка написана мовою програмування

системи MatLab. Програмний код цієї функції наведений у додатку К, а особливості її роботи були описані у підрозділі 2.5.8. Модульний принцип побудови із параметричним викликом функцій, притаманній мові програмування системи науково-технічних розрахунків MatLab, дозволяє використовувати написану функцію **Bin_Division** для формування різних типів завадостійких кодів та декодування їхніх послідовностей, зокрема, циклічних кодів, кодів Файра та БЧХ. Для формування кодових послідовностей систематичних кодів БЧХ у програмі використаний алгоритм дописування нулів, ділення на твірний поліном та додавання отриманої остачі. Цей алгоритм також є типовим для формування всіх систематичних циклічних кодів та був описаний у підрозділі 2.5.5. Тому для виконання операції формування кодів БЧХ також використовується функція **Bin_Division**.

На алгоритмі, наведеному на рис. 3.28, зсув вхідної кодової комбінації на одну позицію ліворуч позначено як $\overleftarrow{v_{in}}$, а зворотний зсув на i позицій праворуч – як $\overrightarrow{v_{in}}^{i}$.

У додатку М наведені також результати тестування програми **BCH**. Із наведених даних тестування зрозуміло, що для формування коду БЧХ (15, 7) може бути використана будь-яка вхідна послідовність нулів та одиниць довжиною від двох до сьоми розрядів, а для формування коду БЧХ (15, 5) – будь-яка послідовність довжиною від двох до п’яти розрядів. Замість бітів, яких не вистачає, до вхідної послідовності зліва автоматично дописуються нулі. Якщо виконується операція декодування, вхідна послідовність, незалежно від типу коду БЧХ, має бути п’ятнадцятирозрядною. Оскільки під час виконання процедури кодування сформована послідовність коду БЧХ є вихідним вектором, ця послідовність може бути декодована у разі запуску програми **BCH** з параметром **nop** = 2.

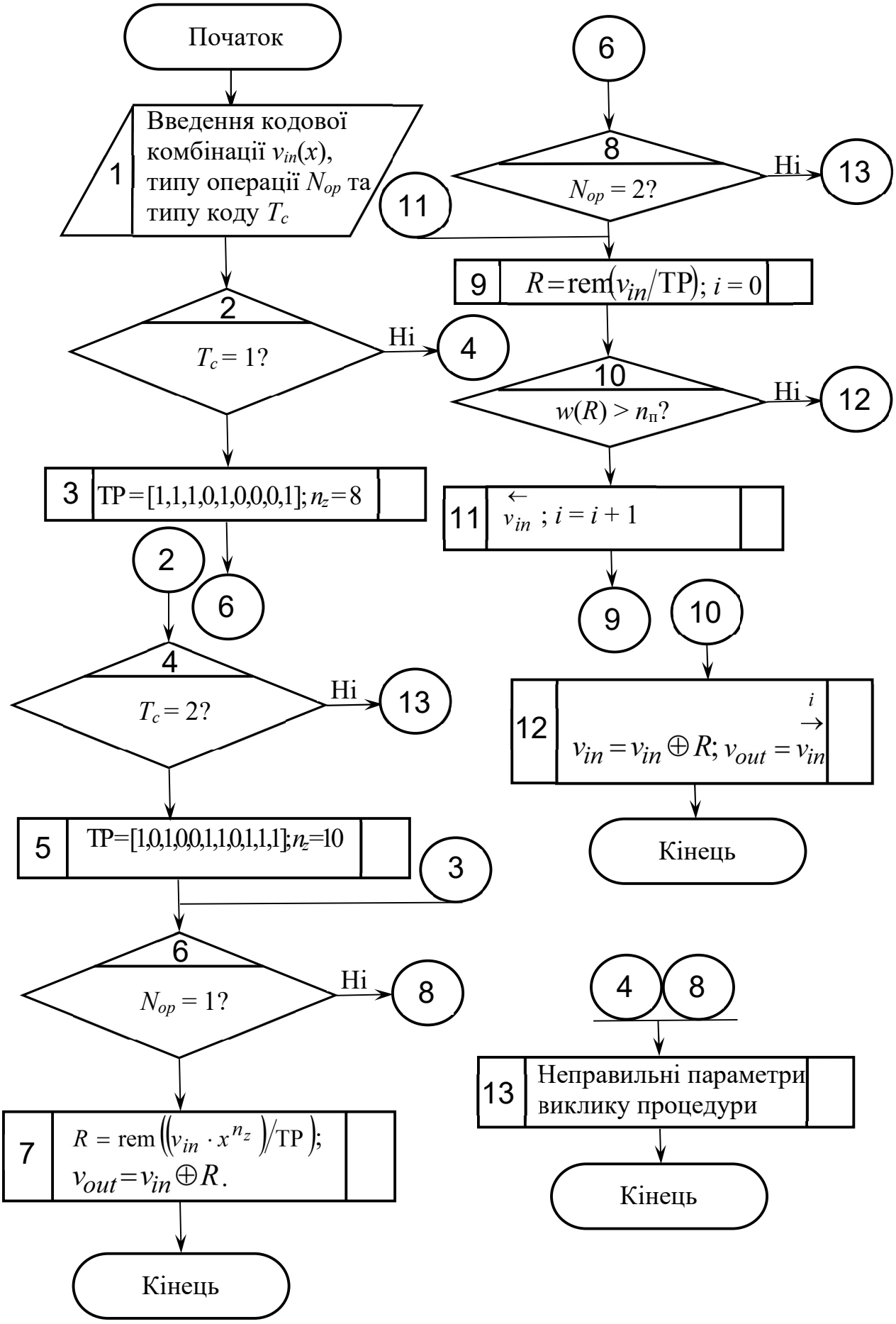


Рис. 3.28 Узагальнений алгоритм роботи програми ВСН

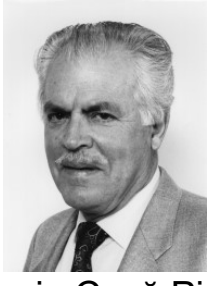
Також із результатів тестування програми видно, що код БЧХ (15, 7) виправляє подвійні помилки, а код БЧХ (15, 5) – потрійні помилки. Якщо кількість помилкових розрядів є більшою, виникає помилка декодування, яка обумовлена переходом до іншої дозволеної кодової комбінації. У разі, якщо помилкові розряди знайдені, на екран виводиться повідомлення про номери цих розрядів та вірна кодова комбінація із виправленими розрядами. Крім цього, формується вектор закодованої бітової послідовності, якій може бути використаний для подальших розрахунків складної кодувальної системи.

Коди БЧХ є одними із самих поширених типів завадостійких групових кодів, які використовуються у сучасній кодувальній електронній апаратурі, у системах зв'язку та у комп'ютерних системах [52, 53, 80]. Важливим різновидом кодів БЧХ, який у навчальній літературі розглядається як його окремий випадок, є код Ріда – Соломона [62, 63, 81]. Цей тип групових кодів розглядатиметься у наступному підрозділі.

3.4 Коди Ріда – Соломона

3.4.1 Загальне визначення кодів Ріда – Соломона як різновидів кодів Боуза – Чоудхурі – Хоквінгема

Коди Ріда – Соломона, або, скорочено, РС-коди (англійський термін – Reed – Solomon code, R-S code), є одним із різновидів багатопозиційних кодів БЧХ, розглянутих у підрозділі 3.3.3 [56, 57, 62, 63]. Ці коди вперше були запропоновані співробітниками лабораторії імені Авраама Лінкольна Масачусетського технологічного інституту Ірвіном Рідом та Густовом Соломоном у 1960 році.



Ірвін Стой Рід
(1923 – 2012)

Густав Соломон
(1930 —1996)

Особливості реалізації РС-кодів полягають у тому, що вони завжди будуються не над двійковими послідовностями, а над послідовностями із кількістю бітів m. Тобто, на відміну від розглянутих у попередніх підрозділах звичайного циклічного коду та двійкових БЧХ, у РС-кодах операції виконуються над полем Галуа $GF(2^m)$, де m – кількість бітів у символах, які кодуються [33, 56, 57, 81].

Для декодування кодів РС використовуються алгоритми ПГЦ та Берлекемпа – Мессі, розглянуті у підрозділі 3.3.4. Саме починаючи з робіт Е. Бепрлекемпа та Дж. Мессі, опублікованих у 1969 р., де був розглянутий ефективний алгоритм декодування багатопозиційних кодів БЧХ та способи його апаратної реалізації, РС-коди починають ефективно використовуватися в електронній апаратурі. А починаючи з вісімдесятих років минулого століття коди РС вже дуже широко використовуються в багатьох стандартах цифрової електронної та апаратури, зокрема, у стандартах обчислювальної техніки та телекомунікаційних мереж [56, 57]. Наприклад, з 1982 року РС-коди покладені в основу шифрування інформації на компакт-дисках. Сьогодні головним сферами застосування РС-кодів є наступні [56, 57].

1. Зовнішні пристрої пам’яті на магнітних стрічках для ЕОМ.

2. Контролери оперативної пам’яті персональних комп’ютерів.

3. Комунікаційні пристрої комп’ютерних мереж, зокрема модеми та комутатори.

4. Контролери жорстких дисків персональних комп’ютерів.

5. Шифрування цифрової інформації в системах записування та читання оптичних дисків стандартів CD-ROM, CD-RW, DVD та Blue Ray.

Надамо визначення коду РС, у якому опишемо загальний спосіб його формування.

Визначення 3.2. Кодом РС називається БЧХ-код із твірним поліномом, який задається логічним виразом [62, 63]:

$$g(x)=\prod_{i=1}^{2\cdot t}\left(x-\alpha^{i}\right), \tag{3.199}$$

де q – основа групи Галуа $GF(q^m)$, α – примітивний елемент цього поля, t – кількість помилок, які виправляються.

За умови $q = 2$ співвідношення (3.199) переписується наступним чином [62, 63]:

$$g(x) = \prod_{i=1}^{2 \cdot t} \left(x - 2^i \right), \tag{3.200}$$

Найчастіше для формування РС-кодів використовують поля Галуа $GF(2^m)$ та формулу (3.200), що відповідає кодуванню бітових послідовностей. Наприклад, поле Галуа $GF(2^8)$ відповідає кодуванню одного байта, у цьому випадку кодуються натуральні числа від 0 до 255.

3.4.2 Параметри кодів Ріда – Соломона

Головними параметрами кодів РС, як різновиду кодів БЧХ, є наступні [62, 63].

1. Кількість перевірочних символів r відповідає степені породжувального поліному.

$$r = 2 \cdot t \ = d_{\min} - 1. \tag{3.201}$$

2. Загальна кількість символів коду n визначається із співвідношення:

$$n = k + d_{\min} = 2 \cdot t + k, \tag{3.202}$$

де k – кількість інформаційних символів коду.

Загалом співвідношення (3.201), (3.202) є загальними для кодів БЧХ та відповідають теоремі 3.3 та співвідношенню (3.23). Відповідно, параметри надлишковості для РС-кодів розраховуються за співвідношеннями (3.24).

Єдиною відмінністю багатопозиційних кодів БЧХ над полем Галуа $GF(2^m)$ від двійкових кодів, які розглядалися у цьому посібнику раніше, є визначення мінімальної кодової відстані $d_{\min}$. Надамо відповідні визначення [62, 63].

Визначення 3.3. Кодовою відстанню між двома багатопозиційними кодовими комбінаціями називається кількість розрядів у них, які відрізняються.

Наприклад, припустимо, що розглядаються кодові комбінації шістнадцяткового коду, символи якого кодуються числами від 0 до 15. Будемо

вважати, що це кодові комбінації [9, 3, 12] та [9, 5, 12]. Згідно із визначенням 3.3 зрозуміло, що кодова відстань між ними становить $d = 1$. А для кодових комбінацій [9, 3, 12] та [3, 9, 12] кодова відстань дорівнює $d = 2$.

Визначення 3.4. Мінімальною кодовою відстанню багатопозиційного коду називається мінімальна можлива кодова відстань між двома дозволеними кодовими комбінаціями.

Розглянемо відповідну теорему теорії кодування, яка визначає зв'язок між параметрами РС-кодів [62, 63].

Теорема 3.10. Код РС має мінімальну кодову відстань

$$d_{\min} = n - k = 2 \cdot t + 1 \qquad (3.203)$$

і завжди є циклічним кодом із максимальною мінімальною кодовою відстанню.

Зрозуміло, що співвідношення (3.203) є наслідком співвідношення (3.202), яке випливає із співвідношення (3.23) для кодів БЧХ та теореми 3.3.

Слід зазначити, що, згідно із співвідношенням (3.200), кількість контрольних символів у РС-кодах пропорціональна $2 \cdot t$. З точки зору логіки організації коду це означає, що один контрольний символ використовується для виявлення позиції помилки, а другий – для визначення точного значення спотвореного символу [56, 57].

Одним із головних понять теорії кодування, пов'язаним із груповими РС-кодами, є поняття про стирання символів. Розглянемо відповідні теоретичні положення [56, 57].

Загалом завдання, які виникають у теорії сигналів для цифрових обчислювальних систем та систем зв'язку, поділяються на дві групи. До першого типу завдань відносяться завдання створення оптимальних рішень на виході декодера з точки зору принципу формування відповідного коду, зазвичай такі рішення формуються на апаратному рівні. Наприклад, у другому розділі розглядалися відповідні декодувальні схеми для кодів Хеммінга та циклічних кодів. Така ситуація є типовою для цифрових систем зв'язку із пам'яттю, наприклад, для систем запису інформації на магнітні або оптичні

носії. За таких умов виникнення помилки є випадковим і пов’язано насамперед із випадковими факторами, зокрема, із шумами у системі. Такий процес у теорії інформації називається стиранням символів. Надамо відповідне визначення [56, 57].

Визначення 3.5. Стиранням (англійський термін – deletion) символів у кодовій комбінації називається відсутність цих символів через вплив завад та спотворення інформації.

Для виправлення стертих символів у коротких пакетах у системах зв’язку використовують РС-коди.

Завдання другого типу у теорії кодування виникають за умови наявності шумів. У цьому разі зазвичай канал зв’язку можна описати як систему із білим гауссівським шумом, в якій час від часу виникають шумові сплески. Відповідні інформаційні моделі розглядалися у четвертому розділі першої частини посібника [1]. За таких умов розв’язування задачі декодування є простішим, оскільки символи, які спотворені завадою, можна оголосити стертими та поновити їх з використанням методів завадостійкого кодування. У разі використання РС-кодів можна виправляти помилки заданої кратності, залежно від коректувальної здатності коду. Для цього вводиться відповідне поняття пакетів помилок. Надамо відповідне визначення.

Визначення 3.6. Пакетом помилок (англійський термін – error burst) називається послідовність стертих символів заданої кратності.

Важливими параметрами групових завадостійких кодів, які дозволяють виправляти пакетні помилки, є довжина пакетів B, періодичність слідування пакетів помилок P та інтенсивність пакетів δ. Надамо відповідні визначення [56, 57].

Визначення 3.7. Довжиною пакетів помилок B називається кількість символів коду, у яких визначаються помилки заданої кратності.

Згідно із властивостями РС-кодів зрозуміло, що для коректної роботи коду необхідним є виконання умови

$$B > t. \tag{3.204}$$

Визначення 3.8. Періодичністю пакетів помилок P називається кількість символів коду, які слідують між двома спотвореними пакетами та вважаються неспотвореними.

Визначення 3.9. Інтенсивністю пакетів помилок називається відношення довжини пакетів до їхнього періоду, тобто:

$$\delta = \frac{B}{P}. \tag{3.205}$$

Зрозуміло, що головні труднощі виявлення помилок у групових кодах пов'язані із випадковістю процесу стирань. Тому зазвичай в реальних кодувальних системах довжина спотвореного пакету B вважає фіксованою величиною, а інтенсивність пакетів δ – усередненою величиною із експоненціальним розподілом часу між сусідніми пакетами [56, 57].

Імовірність появи помилки к декодованому повідомленні для кодів Ріда – Соломона можна обчислити через співвідношення [33, 52, 62, 63]:

$$P_E = \frac{1}{2^m - 1} \sum_{j=t+1}^{2^m-1} j C_{2^m-1}^{j} p^j p^{2^m-1-j}, \tag{3.206}$$

де p – імовірність спотворення одного символу, P_E – імовірність невиявленої помилки, $C_{2^m-1}^{j}$ – кількість сполучень із $2m - 1$ по j, яка обчислюється наступним чином [33, 52, 62, 63]:

$$C_{2^m-1}^{j} = \frac{(2m-1)!}{j!(2m-1-j)!}. \tag{3.207}$$

Дані, наведені у підручниках [33, 52, 62, 63], свідчать про те, що у реальних системах зв'язку, де використовуються РС-коди, за умови $n = 31$ та $p = 10^{-3}$, для $t = 1$ $P_E = 10^{-5}$, а для $t = 2$ – $P_E = 10^{-7}$. Для тієї ж самої системи, за умови $t = 8$ та $p = 5 \cdot 10^{-1}$, $P_E = 10^{-6}$. Особливо ефективно використовуються РС-коди у каналах цифрового зв'язку для боротьби із імпульсними завадами. Наприклад, розглянемо РС-код із параметрами $n = 255$, $k = 247$, $m = 8$, та припустимо, що з 255 біт 25 біт передаються в умовах дії сильної імпульсної завади. Зрозуміло, що час дії завади складає біля 10% від часу передавання сигналу. Крім того, ясно, що оскільки 25 біт – це більше, ніж 3 байти, у

повідомленні, яке передається, буде спотворено 4 байти. Але перевага кодів Ріда – Соломона полягає у тому, що за таких умов передавання інформації на приймальній стороні будуть виправлені всі 4 байти повідомлення, не залежно від того, скільки біт було спотворено, 1, 2, 3 або 4. І саме ця властивість вигідно відрізняє РС-коди від звичайних цифрових кодів. Теоретичні оцінки коректувальної здатності кодів Ріда – Соломона будуть наведені у підрозділі 3.4.12. Більш досконало параметри РС-кодів розглядаються та аналізуються у підручниках [33, 81].

3.4.3 Відображення кодів Ріда – Соломона на двійкові коди

Якщо коди РС формуються над полем Галуа $GF(2^m)$, вони завжди можуть бути відображені на двійкові послідовності [56, 57]. Нехай $\xi_1, \xi_2, ..., \xi_m$ – базис елементів поля Галуа $GF(2^m)$ над полем $GF(2)$. Тоді, якщо $\beta = \sum_{i=1}^{m} b_i \xi_i$ – довільний елемент поля $GF(2^m)$, а b_i – елемент поля $GF(2)$, то існує однозначне гомоморфне відображення $\beta \to \{b_1, b_2 \ldots b_m\}$.

Розглянуте відображення $\beta \to \{b_1, b_2 \ldots b_m\}$ переводить лінійні коди в нелінійні, проте циклічна структура РС-коду зазвичай зберігається [56, 57].

Розглянемо приклад формування РС-коду над полем Галуа $GF(4)$ та відображення елементів цього коду на поле Галуа $GF(2)$ [56, 57].

Приклад 3.23. Побудувати РС-код (3, 2) над полем Галуа $GF(4)$ та знайти відображення елементів цього коду на поле $GF(2)$.

Для коду РС (3, 2) породжувальний поліном записується у вигляді $g(x) = x - \alpha$. Породжувальна матриця для такого коду має вигляд:

$$G(3,2) = \begin{bmatrix} \alpha & 1 & 0 \\ 0 & \alpha & 1 \end{bmatrix}.$$

Тоді код РС (3, 2) містить 16 елементів, які наведені у таблиці 3.11.

Таблиця 3.11 – Послідовності коду РС(3, 2) із примітивним елементом α для прикладу 3.23

№	Послідовність символів	№	Послідовність символів	№	Послідовність символів	№	Послідовність символів
0	0 0 0	4	0 α 1	8	$0\ \alpha\ \alpha^2$	12	$0\ 1\ \alpha^2$
1	α 1 0	5	$\alpha\ \alpha^2\ 1$	9	α α α	13	$\alpha\ 0\ \alpha^2$
2	$\alpha^2\ \alpha\ 0$	6	$\alpha^2\ 0\ 1$	10	$\alpha^2\ 1\ \alpha$	14	$\alpha^2\ \alpha^2\ \alpha^2$
3	$1\ \alpha^2\ 0$	7	1 1 1	11	1 0 α	15	$1\ \alpha\ \alpha^2$

У подальших міркуваннях будемо вважати, що 0 → 00, 1 → 10, α → 01, α^2 → 11. Тоді РС-код (3, 2) над полем Галуа *GF*(4) відображається на двійковий код РС (6, 4) над полем Галуа *GF*(2). Відповідні елементи цього коду наведені у таблиці 3.12.

Таблиця 3.12 – Послідовності двійкового коду РС(6, 4) для прикладу 3.23

№	Послідовність символів	№	Послідовність символів	№	Послідовність символів	№	Послідовність символів
0	0 0 0 0 0 0	4	0 0 0 1 1 0	8	0 0 1 1 0 1	12	0 0 1 0 1 1
1	0 1 1 0 0 0	5	0 1 1 1 1 0	9	0 1 0 1 0 1	13	0 1 0 0 1 1
2	1 1 0 1 0 0	6	1 1 0 0 1 0	10	1 1 1 0 0 1	14	1 1 1 1 1 1
3	1 0 1 1 0 0	7	1 0 1 0 1 0	11	1 0 0 0 0 1	15	1 0 0 1 1 1

Розглянемо інший приклад [56, 57].

Приклад 3.24. Побудувати РС-код (7, 5) над полем Галуа *GF*(8) із кодовою відстанню $d_{\min} = 3$ та знайти відображення елементів цього коду на поле *GF*(2).

Поле Галуа *GF*(8), побудоване за модулем $\Pi(\alpha) = \alpha^3 + \alpha + 1$, містить такі вісім елементів:

0 0 0 = 0; 1 0 0 = 1; 0 1 0 = α; 0 0 1 = α^2;

1 1 0 = α^3; 0 1 1 = α^4; 1 1 1 = α^5; 1 0 1 = α^6.

Як базис для формування елементів поля Галуа $GF(2)$ із елементів поля $GF(8)$ використаємо елементи 1, α^3 та α^6. Відповідно маємо:

$$\beta = b_1(1,0,1) + b_2(0,1,0) + b_3(1,0,1). \qquad (3.208)$$

Враховуючи співвідношення (3.206), отримуємо наступне відображення:

$0 \rightarrow 000$; $\alpha \rightarrow 010$; $\alpha^2 \rightarrow 101$; $\alpha^3 \rightarrow 110$;

$\alpha^4 \rightarrow 111$; $\alpha^5 \rightarrow 100$; $\alpha^5 \rightarrow 100$; $\alpha^6 \rightarrow 001$.

Породжувальний поліном можна записати у вигляді:

$$g(x) = (x + \alpha^5)\cdot(x + \alpha^6) = x^2 + \alpha\cdot x + \alpha^4.$$

За умови такої побудови код РС (7, 5) над полем Галуа $GF(8)$ відображується на двійковий циклічний код (21, 15) із твірним поліномом

$$g(x) = x^6 + x^4 + x^2 + x + 1.$$

Мінімальна кодова відстань такого двійкового коду становить $d_{\min} = 3$. Розглянутий приклад є єдиним відомим прикладом відображення групового коду РС на двійковий циклічний код [56, 57].

Приклад формування двійкової послідовності для коду Ріда – Соломона (7, 3) у полі Галуа $GF(2^3)$ розглядатиметься у підрозділі 3.4.9.

3.4.4 Основні алгебраїчні та поліноміальні операції у полях Галуа $GF(2^m)$

Перед вивченням цього підрозділу необхідно повторити розділи 2 та 3 та підрозділ 4.3 другої частини посібника, а також підрозділ 2.5 цієї частини посібника

3.4.4.1 Алгебраїчні операції над елементами полів Галуа $GF(2^m)$ та їхня програмна реалізація у системі MatLab

Головна особливість побудови кодів РС над полем Галуа $GF(2^m)$ полягає у тому, що, як і в циклічних кодах, всі поліноміальні операції здійснюються за модулем 2. Проте, якщо для двійкових циклічних кодів та кодів БЧХ, розглянутих у підрозділі 3.3, ця операція є вкрай простою та зрозумілою, для багаторозрядних чисел, записаних у полі Галуа $GF(2^m)$, ця операція є не настільки наочною та потребує додаткового розтлумачення. Для операцій додавання та віднімання ця складність пов'язана лише із тим, що числа, які сумуються, записані у десятковому, а не у двійковому форматі. Тому для

здійснення операції «виключного або» під час сумування двох чисел необхідно спочатку перевести число у двійковий формат. Це перетворення є подібним до відображення кодів PC на двійкові коди, яке розглядалася у попередньому підрозділі. Розглянемо приклади виконання операції сумування через функцію «виключного або» у полях Галуа $GF(2^m)$.

Приклад 3.25. Знайти результат операції $5 \oplus 4$ у полі Галуа $GF(2^3)$.

Зрозуміло, що $5_8 = 101_2$, а $4_8 = 100_2$.

Тоді: $5_8 \oplus 4_8 = 101_2 \oplus 100_2 = 001_2 = 1_8$.

Приклад 3.26. Знайти результат операції $51 \oplus 149$ у полі Галуа $GF(2^8)$.

Зрозуміло, що $51_{10} = 00110011_2$, а $149_{10} = 10010101_2$. Тоді:

$$51_{10} \oplus 149_{10} = 00110011_2 \oplus 10010101_2 = 10100110_2 = 166_{10}.$$

Більш складними операціями з алгебраїчної точки зору є множення та ділення елементів поля Галуа $GF(2^m)$. Вони виконується через відоме із елементарної математики правила сумування та віднімання логарифмів, яке свідчить про те, що логарифм від добутку двох чисел дорівнює сумі логарифмів цих чисел, а логарифм від частки двох чисел – їх різниці [51].

Розглянемо відповідні властивості, які визначають порядок дій для множення та ділення елементів поля Галуа $GF(2^m)$.

Визначення 3.10. Алгебраїчну операцію множення для елементів поля Галуа $GF(2^m)$ можна звести до трьох таких послідовних операцій.

1. Якщо один із множників дорівнює 0, вважається, що результатом множення є 0. В іншому випадку в полі Галуа $GF(2^m)$ обчислюються двійкові логарифми чисел, які множаться.

2. Сумування отриманих результатів логарифмування за правилами звичайної арифметики. Якщо результат цього сумування перевищує $2^m - 1$, необхідно відняти число $2^m - 1$ від отриманого результату.

3. Піднести число 2 до степені числа, яке було отримане у попередніх розрахунках.

Визначення 3.11. Алгебраїчну операцію ділення для елементів поля Галуа $GF(2^m)$ здійснюється зворотним чином до операції множення. Тобто, послідовність елементарних операцій є наступною.

1. Якщо ділене дорівнює 0, результат ділення вважається рівним 0. Якщо дільник дорівнює 0, результат ділення вважається невизначеним. В іншому

випадку в полі Галуа $GF(2^m)$ обчислюються двійкові логарифми чисел, які діляться.

2. Віднімання від логарифму діленого логарифму дільника за правилами звичайної арифметики. Якщо результат цього віднімання є негативним, необхідно додати до отриманого результату число $2^m - 1$.

3. Піднести число 2 до степені числа, яке було отримане у попередніх розрахунках.

У розглянутих алгоритмах множення та ділення елементів поля Галуа $GF(2^m)$ найбільш складною є операція піднесення числа 2 до елементів поля Галуа i, тобто, обчислення значень функції 2^i. Якщо ця операція реалізована, для обчислення логарифмів чисел розглядається зворотна операція. Тобто, за умови відомої таблиці значень $j = 2^i$ необхідно знайти значення $i = \log_2(j)$. Оскільки, згідно із теорією алгебраїчних операцій у полях Галуа $GF(2^m)$, операція піднесення до степені є зімкненою на множені елементів поля, операція взяття логарифму також існує і є гомоморфною [48, 52, 55, 62, 63]. Відповідний математичний апарат для алгебраїчних операцій над елементами полів Галуа розглядався у другому та третьому розділах другої частини посібника [48]. Наприклад, значення $j = 2^i$ для елементів поля Галуа $GF(256)$ можуть бути обчислені через наступне рекурентне співвідношення [52, 63, 81]:

$$j(i) = 2^i = \begin{cases} 1, & \text{якщо } i = 0; \\ 0, & \text{якщо } i = 255; \\ 2 \cdot 2^{i-1}, & \text{якщо } 0 < j(i-1) < 128; \\ \left(2 \cdot 2^{i-1}\right) \oplus 285, & \text{якщо } 128 \le j(i-1) < 254, \end{cases} =$$

$$= \begin{cases} 1, & \text{якщо } i = 0; \\ 0, & \text{якщо } i = 255; \\ \overleftarrow{2^{i-1}}, & \text{якщо } 1 < j(i-1) < 128; \\ \overleftarrow{2^{i-1}} \oplus 285, & \text{якщо } 128 \le j(i-1) < 254. \end{cases} \qquad (3.209)$$

У формулі (3.209) знак $\overleftarrow{2^{i-1}}$ означає зсув двійкового числа 2^{i-1} на одну позицію ліворуч, а число 285 відповідає незвідному поліному $x^8 + x^4 + x^3 + x^2 + 1$. Логіку обчислення степеневої функції для елементів поля Галуа $GF(256)$ можна пояснити наступним чином. Значення $2^0 = 1$ та $2^{255} = 0$ задаються

аксіоматично. Після цього для значень $1 < i < 128$ елемент l_i обчислюється через елемент $l_{i\text{-}1}$ шляхом його множення на 2, а для значень $128 \le l_i < 254$ до результату множення на 2 необхідно додати за модулем 2 число 285, яке відповідає твірному поліному заданого поля Галуа $GF(256)$, $g(x) = x^8 + x^4 + x^3 + x^2 + 1$. Дійсно, подання цього поліному у формі двійкового числа має вигляд 10001111_2, що відповідає десятковому числу 285_{10}. Зрозуміло, що якщо число 2^{i-1} подано у двійковій формі, його множення на 2 є еквівалентним арифметичному зсуву бітів числа на одну позицію ліворуч.

Співвідношення (3.209) записано для виконання операції піднесення до степеня для елементів поля Галуа $GF(256)$, що відповідає кодуванню одного байту інформації. Проте зрозуміло, що для інших полів Галуа класу $GF(2^m)$ аналогічне співвідношення буде відрізнятися лише порядком поля m та твірним поліномом $g_m(x)$. У загальному вигляді співвідношення (3.208) можна переписати наступним чином [52, 63]:

$$j(i) = 2^i = \begin{cases} 1, & \text{якщо } i = 0; \\ 0, & \text{якщо } i = 2^m; \\ 2 \cdot 2^{i-1}, & \text{якщо } 0 < j(i-1) < 2^{m-1}; \\ \left(2 \cdot 2^{i-1}\right) \oplus c_m, & \text{якщо } 2^{m-1} \le j(i-1) < 2^m - 1, \end{cases} =$$

$$= \begin{cases} 1, & \text{якщо } i = 0; \\ 0, & \text{якщо } i = 2^m; \\ \overleftarrow{2^{i-1}}, & \text{якщо } 1 < j(i-1) < 2^{m-1}; \\ \overleftarrow{2^{i-1}} \oplus c_m, & \text{якщо } 2^{m-1} \le j(i-1) < 2^m - 1, \end{cases} \qquad (3.210)$$

де c_m – двійкове число, яке відповідає твірному поліному $g_m(x)$ для поля Галуа $GF(2^m)$. Поліноміальне та числове подання для деяких твірних поліномів над полями Галуа $GF(2^m)$ за умови $3 \le m \le 24$ наведене у таблиці 3.13 [33]. Зрозуміло, що ці поліноми для значень $3 \le m \le 8$ цілком відповідають деяким незвідним поліномам, наведеним у додатку І.

Таблиця 3.13 – Деякі незвідні твірні поліноми для полів Галуа $GF(2^m)$ для різних значень порядку поля m [33]

m	Поліном	Числове подання	m	Поліном	Числове подання
3	$x^3 + x + 1$	$1011_2 = 11_{10}$	14	$x^{14} + x^{10} + x^6 + x + 1$	$100010001000011_2 = 17475_{10}$
4	$x^4 + x + 1$	$10011_2 = 19_{10}$	15	$x^{15} + x + 1$	$1000000000000011_2 = 32771_{10}$
5	$x^5 + x^2 + 1$	$100101_2 = 37_{10}$	16	$x^{16} + x^{12} + x^3 + x + 1$	$1000100000000011011_2 = 69643_{10}$
6	$x^6 + x + 1$	$1000011_2 = 67_{10}$	17	$x^{17} + x^3 + 1$	$10000000000000001001_2 = 131081_{10}$
7	$x^7 + x^3 + 1$	$10001001_2 = 137_{10}$	18	$x^{18} + x^7 + 1$	$10000000000100000001_2 = 262273_{10}$
8	$x^8 + x^4 + x^3 + x^2 + 1$	$100011101_2 = 285_{10}$	19	$x^{19} + x^5 + x^2 + x + 1$	$10000000000000100111_2 = 524327_{10}$
9	$x^9 + x^4 + 1$	$1000010001_2 = 529_{10}$	20	$x^{20} + x^3 + 1$	$10000000000000001001_2 = 1048585_{10}$
10	$x^{10} + x^3 + 1$	$10000001001_2 = 1033_{10}$	21	$x^{21} + x^2 + 1$	$10000000000000000101_2 = 2097157_{10}$
11	$x^{11} + x^2 + 1$	$100000000101_2 = 2053_{10}$	22	$x^{22} + x + 1$	$10000000000000000011_2 = 4194307_{10}$
12	$x^{12} + x^6 + x^4 + x + 1$	$1000001010011_2 = 4179_{10}$	23	$x^{23} + x^5 + 1$	$10000000000000000100001_2 = 8388641_{10}$
13	$x^{13} + x^4 + x^3 + x + 1$	$10000000011011_2 = 8219_{10}$	24	$x^{24} + x^7 + x^2 + x + 1$	$100000000000000010000111_2 = 16777351_{10}$

Розраховані значення функції 2^i для поля Галуа $GF(256)$ наведені у таблиці 3.14, а значення зворотної функції $\log_2(i)$ для цього самого поля – у таблиці 3.15 [81].

Таблиця 3.14 – Значення функції 2^i для елементів поля Галуа *GF*(256)

+	0	1	2	3	4	5	6	7	8	9
0	1	2	4	8	16	32	64	128	29	58
10	116	232	205	135	19	38	76	152	45	90
20	180	117	234	201	143	3	6	12	24	48
30	96	192	157	39	78	156	37	74	148	53
40	106	212	181	119	238	193	159	35	70	140
50	5	10	20	40	80	160	93	186	105	210
60	185	111	222	161	95	190	97	194	153	47
70	94	188	101	202	137	15	30	60	120	240
80	253	231	211	187	107	214	177	127	254	225
90	223	163	91	182	113	226	217	175	67	134
100	17	34	68	136	13	26	52	104	208	189
110	103	206	129	31	62	124	248	237	199	147
120	59	118	236	197	151	51	102	204	133	23
130	46	92	184	109	218	169	79	158	33	66
140	132	21	42	84	168	77	154	41	82	164
150	85	170	73	146	57	114	228	213	183	115
160	230	209	191	99	198	145	63	126	252	229
170	215	179	123	246	241	255	227	219	171	75
180	150	49	98	196	149	55	110	220	165	87
190	174	65	130	25	50	100	200	141	7	14
200	28	56	112	224	221	167	83	166	81	162
210	89	178	121	242	249	239	195	155	43	86
220	172	69	138	9	18	36	72	144	61	122
230	244	245	247	243	251	235	203	139	11	22
240	44	88	176	125	250	233	207	131	27	54
250	108	216	173	71	142	0	–	–	–	–

Таблиця 3.15 – Значення функції $\log_2(i)$ для елементів поля Галуа *GF*(256)

+	0	1	2	3	4	5	6	7	8	9
0	255	0	1	25	2	50	26	198	3	223
10	51	238	27	104	199	75	4	100	224	14
20	52	141	239	129	28	193	105	248	200	8
30	76	113	5	138	101	47	225	36	15	33
40	53	147	142	218	240	18	130	69	29	181
50	194	125	106	39	249	185	201	154	9	120
60	77	228	114	166	6	191	139	98	102	221
70	48	253	226	152	37	179	16	145	34	136
80	54	208	148	206	143	150	219	189	241	210
90	19	92	131	56	70	64	30	66	182	163
100	195	72	126	110	107	58	40	84	250	133
110	186	61	202	94	155	159	10	21	121	43
120	78	212	229	172	115	243	167	87	7	112
130	192	247	140	128	99	13	103	74	222	237
140	49	197	254	24	227	165	153	119	38	184
150	180	124	17	68	146	217	35	32	137	46
160	55	63	209	91	149	188	207	205	144	135
170	151	178	220	252	190	97	242	86	211	171
180	20	42	93	158	132	60	57	83	71	109
190	65	162	31	45	67	216	183	123	164	118
200	196	23	73	236	127	12	111	246	108	161
210	59	82	41	157	85	170	251	96	134	177
220	187	204	62	90	203	89	95	176	156	169
230	160	81	11	245	22	235	122	117	44	215
240	79	174	213	233	230	231	173	232	116	214
250	244	234	168	80	88	175	–	–	–	–

Важливою властивістю полів Галуа є їхня мультиплікативність, яка випливає із визначень 3.10 та 3.11. Як відомо, сутність цієї властивості полягає у тому, що у полі Галуа $GF(2^m)$ існує нульовий елемент 0. Множення на цей елемент будь-якого елемента a завжди дає нульовий результат, але якщо обидва множники, a та b, не є нулевими елементами, результат множення цих елементів завжди буде не 0 [48]. Формулу для множення двох елементів поля Галуа $GF(2^m)$, згідно із визначенням 3.10, можна записати у вигляді [81]:

$$a \cdot b = \begin{cases} 0, & ((a=0) \vee (b=0)); \\ 2^{\log_2(a)+\log_2(b)}, & \left((\log_2(a)+\log_2(b)) < 2^m - 1\right) \wedge (a \neq 0) \wedge (b \neq 0); \\ 2^{\log_2(a)+\log_2(b)-(2^m-1)}, & \left((\log_2(a)+\log_2(b)) \geq 2^m - 1\right) \wedge (a \neq 0) \wedge (b \neq 0). \end{cases} \quad (3.211)$$

Відповідно, для операції ділення, згідно із визначенням 3.11:

$$\frac{a}{b} = \begin{cases} 0, & ((a=0) \wedge (b \neq 0)) \\ \text{Помилка ділення}, & (b=0) \\ 2^{\log_2(a)-\log_2(b)}, & ((\log_2(a)-\log_2(b)) \geq 0) \wedge (a \neq 0) \wedge (b \neq 0); \\ 2^{\log_2(a)-\log_2(b)+(2^m-1)}, & ((\log_2(a)-\log_2(b)) < 0) \wedge (a \neq 0) \wedge (b \neq 0). \end{cases} \quad (3.212)$$

Мультиплікативність полів Галуа $GF(2^m)$ є важливою аксіомою теорії груп [49, 52, 53, 55], яка також широко використовується у сучасній теорії кодування цифрових сигналів для формування групових кодів. Річ у тому, що саме нульовий елемент поля завжди використовується як індикатор відсутності помилки у коді, тому під час проведення обчислень правильність виконання арифметичних операцій саме із цим елементом є вкрай важливою.

Дійсно, припустимо, що ми ігнорували правило множення на 0 та виконали цю операціє з використанням третього рядка формули (3.211), тобто. через обчислення логарифмів. У цьому випадку отримуємо наступний результат множення:

$$a \cdot 0 = 2^{\log_2(a)+\log_2(0)} = 2^{\log_2(a)+\left(2^m-1\right)-\left(2^m-1\right)} = 2^{\log_2(a)} = a.$$

Такий результат обчислень є хибним, оскільки під час проведення розрахунків не врахована мультиплікативність поля Галуа $GF(2^m)$ та властивість нульового елементу. Слід відзначити, що у цьому випадку полю Галуа взагалі була б не притаманна не мультиплікативність, оскільки операція

множення стає не інваріантною. Дійсно, за таких умов вірною є тотожність $a \cdot 0 = a \cdot 1 = a$, тобто, для операції множення нульовий елемент ототожнюється з одиничним. Поле Галуа із таким визначенням операції множення не може бути використане для формування групових кодів, оскільки стає неможливим виявлення нулів у рівняннях синдромів помилок, а, як було вказано у підрозділах 3.1 та 3.3, саме наявність нулів у цих рівняннях свідчить про відсутність помилки у коді. Найважливіший висновок із наведених міркувань полягає у тому, що для формування групових кодів, зокрема кодів БЧХ та Ріда – Соломона, можуть бути використані лише мультиплікативні поля Галуа [52, 55, 62, 63, 81].

Розглянемо тепер приклади виконання алгебраїчних операцій піднесення числа 2 до степені та логарифмування для різних полів Галуа $GF(2^m)$ з використанням формул (3.209), (3.210).

Приклад 3.27. Скориставшись формулою (3.210), знайти результати операцій піднесення числа 2 до степені та зворотної операції логарифмування елементів у полях Галуа $GF(2^3)$ та $GF(2^4)$.

Для поля Галуа $GF(2^3)$, згідно із співвідношенням (3.210), для перших трьох елементів 0, 1 та 2, можна записати: $2^0 = 1$, $2^1 = 2$, $2^2 = 4$. Для наступного елементу, $i = 3$, множення попереднього результату, отриманого для $i = 2$, на число 2, дає значення $4 \cdot 2 = 8$, яке виходить за межу множини елементів поля $GF(2^3)$. Відповідно до співвідношення (3.210) це означає, що необхідно до отриманого результату $8_{10} = 1000_2$ додати за модулем два число c_m, яке відповідає твірному поліному. Звертаючись до даних таблиці 3.13, бачимо, що полю Галуа $GF(2^3)$ відповідає незвідний твірний поліном $x^3 + x + 1$, або, у двійковій числовій формі, 1011_2. Виконання цієї операції дає наступний результат:

$$1000_2 \oplus 1011_2 = 0011_2 = 3_{10}.$$

Тобто, результат обчислень за формулою (3.210) для поля Галуа $GF(2^3)$ є наступним:

$$2^3 = 3.$$

Оскільки значення $3 \cdot 2 = 6$ є елементом поля Галуа $GF(2^3)$, яке розглядається, можна записати:

$$2^4 = 6.$$

Обчислимо тепер значення 2^5. У даному випадку $6 \cdot 2 = 12$, а це значення виходить за межі поля, яке розглядається. Тому, зважаючи на те, що $12_{10} = 1100_2$, необхідно додати за модулем 2 до цього двійкового числа значення твірного поліному 1011_2. Результат виконання цієї операції є наступним:

$$2^5 = 1100_2 \oplus 1011_2 = 0111_2 = 7_{10}.$$

Знайдемо тепер у полі Галуа $GF(2^3)$ значення функції 2^i за умови $i = 6$. Множення попереднього отриманого результату на 2 дає значення

$$7 \cdot 2 = 14_{10} = 1110_2,$$

яке знову виходить за межі поля $GF(2^3)$. Тому, згідно із співвідношенням (3.209), необхідно виконати наступну алгебраїчну операцію:

$$1110_2 \oplus 1011_2 = 0101_2 = 5_{10}.$$

Тепер, знову звертаючись до співвідношення (3.210), знаходимо останнє значення $2^7 = 0$, яке є аксіоматичним. Отримані результати для алгебраїчної операції піднесення числа 2 до степені у полі Галуа $GF(2^3)$ зведені у таблиці 3.16.

Отримати значення зворотної функції $\log_2(i)$ дуже просто за значеннями прямої функції 2^i, оскільки у даному випадку пряма функція описується зімкненим та гомоморфним відношенням. Тобто, аналізуючи другий та перший рядок таблиці 3.16, де записані отримані значення функції 2^i, можна побудувати за ними третій рядок, де записані значення зворотної функції $j = \left(2^i\right)^{-1} = \log_2(i)$. Наприклад, якщо необхідно знайти $\log_2(4)$, необхідно знайти значення 4 у другому рядку таблиці 3.16. Тоді число, яке стоїть у першому рядку тієї самої колонки, і буде значенням зворотної функції $\log_2(4)$.

Відповідно, маємо $\log_2(4) = 2$. Таким саме чином знаходяться інші значення логарифмічної функції $\log_2(i)$ у полі Галуа $GF(2^3)$, відповідні результати наведені у третьому рядку таблиці 3.16.

Таблиця 3.16 – Значення функцій 2^i та $\log_2(i)$ для елементів поля Галуа $GF(2^3)$

Функція	Значення i							
	0	1	2	3	4	5	6	7
2^i	1	2	4	3	6	7	5	0
$\log_2(i)$	7	0	1	3	2	6	4	5

Для поля Галуа $GF(2^4)$ для функції 2^i, згідно із співвідношенням (3.210), враховуючи те, що твірний поліном для цього поля, відповідно до даних, наведених у таблиці 3.13, записується у вигляді 10011_2, маємо наступні результати.

1. $2^0 = 1$; $2^1 = 2$; $2^2 = 4$; $2^3 = 8$.

2. $2^4 = (8 \cdot 2) \oplus c_m = 16_{10} \oplus c_m = 10000_2 \oplus 10011_2 = 11_2 = 3$.

3. $2^5 = 3 \cdot 2 = 6$; $2^6 = 6 \cdot 2 = 12$.

4. $2^7 = (12 \cdot 2) \oplus c_m = 24_{10} \oplus c_m = 11000_2 \oplus 10011_2 = 1011_2 = 11_{10}$.

5. $2^8 = (11 \cdot 2) \oplus c_m = 22_{10} \oplus c_m = 10110_2 \oplus 10011_2 = 101_2 = 5_{10}$.

6. $2^9 = 5 \cdot 2 = 10$.

7. $2^{10} = (10 \cdot 2) \oplus c_m = 20_{10} \oplus c_m = 10100_2 \oplus 10011_2 = 111_2 = 7_{10}$.

8. $2^{11} = 7 \cdot 2 = 14$.

9. $2^{12} = (14 \cdot 2) \oplus c_m = 28_{10} \oplus c_m = 11100_2 \oplus 10011_2 = 1111_2 = 15_{10}$.

10. $2^{13} = (15 \cdot 2) \oplus c_m = 30_{10} \oplus c_m = 11110_2 \oplus 10011_2 = 1101_2 = 13_{10}$.

11. $2^{14} = (13 \cdot 2) \oplus c_m = 26_{10} \oplus c_m = 11010_2 \oplus 10011_2 = 1001_2 = 9_{10}$.

12. $2^{15} = 0$.

Наведені вище результати обчислень степеневої функції 2^i, а також значення зворотної функції $\log_2(i)$ для поля Галуа $GF(2^4)$, в узагальненому вигляді представлені у таблиці 3.17.

Таблиця 3.17 – Значення функцій 2^i та $\log_2(i)$ для елементів поля Галуа $GF(2^4)$

Функція	Значення i															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2^i	1	2	4	8	3	6	12	11	5	10	7	14	15	13	9	0
$\log_2(i)$	15	0	1	4	2	8	5	10	3	14	9	7	6	13	11	12

Як було відмічено раніше, властивості групових кодів, зокрема кодів Ріда – Соломона, зберігаються лише для мультиплікативних полів Галуа. Слід відзначити, що для таких полів вкрай важливою також є інша властивість нульового елементу, а саме: він не може бути результатом множення будь-яких двох елементів, якщо жоден із множників не є нулем. І ця властивість арифметичної операції множення для полів Галуа не є настільки тривіальною та зрозумілою, оскільки, на відміну від звичайної арифметики, у полях Галуа $GF(2^m)$ рівняння $2^i = 0$ має розв'язок $i = 2^m - 1$. Тому саме з цієї причини, згідно із формулою (3.211), віднімати $2^m - 1$ від суми логарифмів необхідно починаючи від значення $2^m - 1$, а не 2^m. Наприклад, знайдемо добуток елементів поля Галуа $GF(2^3)$ 3·6, для чого скористаємося таблицею 3.16. Зважаючи на те, що $\log_2(3) = 3$, $\log_2(6) = 4$, сумуємо ці числа за правилами звичайної арифметики та отримуємо $3 + 4 = 7$. А , згідно із таблицею 3.16, у полі Галуа $GF(2^3)$ $2^7 = 0$. Тобто, у разі проведення таких хибних розрахунків отримуємо невірний результат $3 \cdot 6 = 0$, що не відповідає властивості нульового елементу мультиплікативного поля Галуа. Для отримання вірного результату від суми чисел $3 + 4$ необхідно відняти число 7. Відповідно маємо: $3 + 4 - 7 = 0$, $2^0 = 1$, $3 \cdot 6 = 1$.

Також загальновідомо, що у мультиплікативних полях Галуа до будь-якого елементу поля, крім нуля, існують відповідні обернені елементи [48]. Щоб знайти елемент, обернений до заданого, у полі Галуа $GF(2^m)$, скористаємося формулою (3.212), вважаючи, що $(a=1)\wedge(b\neq 0)$. Зрозуміло, що за визначенням операції ділення для нульового елементу оберненого елементу не існує. Тоді згідно із формулою (3.212) маємо [81]:

$$\frac{1}{b} = 2^{(2^m - 1) - \log_2(b)}. \qquad (3.213)$$

Розглянемо тепер деякі інші приклади множення елементів мультиплікативних полів Галуа.

Приклад 3.28. Скориставшись визначенням 3.10, таблицею 3.17 та формулою (3.11), знайти у полі Галуа *GF*(16) добутки наступних чисел:

$3 \cdot 5, 6 \cdot 7, 12 \cdot 13, 11 \cdot 14$ та $14 \cdot 14$.

Згідно з визначенням 3.10, для знаходження добутків двох чисел у полі Галуа *GF*(16) необхідно, за третім рядком таблиці 3.17, знайти логарифми цих чисел та додати отримані значення за правилом додавання звичайної арифметики. Якщо результат цього додавання є більшим за 15, необхідно відняти 15 від отриманого результату, а після цього, за другим рядком таблиці 3.17, обчислити від розрахованого елементу поля i функцію 2^i. Це і буде остаточним результатом множення двох чисел у полі Галуа *GF*(16). Для операції множення елементів заданого поля Галуа, які розглядаються у даному прикладі, відповідно маємо.

1. Множення чисел $3 \cdot 5$.

Перша дія. За таблицею 3.17 знаходимо: $\log_2(3) = 4$; $\log_2(5) = 8$.

Друга дія. $\log_2(3) + \log_2(5) = 8 + 4 = 12$.

Третя дія. За таблицею 3.17 знаходимо: $2^{12} = 15$.

Тобто, остаточний результат для поля Галуа *GF*(16): $3 \cdot 5 = 15$.

2. Множення чисел $6 \cdot 7$.

Перша дія. За таблицею 3.17 знаходимо: $\log_2(6) = 5$; $\log_2(7) = 10$.

Друга дія. $\log_2(6) + \log_2(7) = 5 + 10 = 15$; $15 - 15 = 0$.

Третя дія. За таблицею 3.17 знаходимо: $2^0 = 1$.

Тобто, остаточний результат для поля Галуа *GF*(16): $6 \cdot 7 = 1$.

3. Множення чисел $12 \cdot 13$.

Перша дія. За таблицею 3.17 знаходимо: $\log_2(12) = 6$; $\log_2(13) = 13$.

Друга дія. $\log_2(12) + \log_2(13) = 6 + 13 = 19$; $19 > 15$; $19 - 15 = 4$.

Третя дія. За таблицею 3.17 знаходимо: $2^4 = 3$.

Тобто, остаточний результат для поля Галуа *GF*(16): $12 \cdot 13 = 3$.

4. Множення чисел 11 · 14.

Перша дія. За таблицею 3.17 знаходимо: $\log_2(11) = 7$; $\log_2(14) = 11$.

Друга дія. $\log_2(11) + \log_2(14) = 7 + 11 = 18$; $18 > 15$; $18 - 15 = 3$.

Третя дія. За таблицею 3.17 знаходимо: $2^3 = 8$.

Тобто, остаточний результат для поля Галуа *GF*(16): $11 \cdot 14 = 8$.

5. Множення чисел 14 · 14.

Перша дія. За таблицею 3.17 знаходимо: $\log_2(14) = 11$.

Друга дія. $\log_2(14) + \log_2(14) = 11 + 11 = 22$; $22 > 15$; $22 - 15 = 7$.

Третя дія. За таблицею 3.17 знаходимо: $2^7 = 11$.

Тобто, остаточний результат для поля Галуа *GF*(16): $14 \cdot 14 = 11$.

Приклад 3.29. Скориставшись визначенням 3.10, формулою (3.11) та таблицями 3.14, 3.15, знайти у полі Галуа *GF*(256) добутки наступних чисел: 3 · 17, 6 · 49, 8 · 49 та 25 · 51.

Згідно з визначенням 3.10, для знаходження добутків двох чисел у полі Галуа *GF*(256) необхідно за таблицею 3.15 знайти логарифми цих чисел та додати отримані значення за правилом додавання звичайної арифметики. Якщо результат цього додавання є більшим за 255, необхідно відняти 255 від цього числа, а після цього за таблицею 3.14 обчислити від отриманого результату i функцію 2^i. Це і є остаточний результат множення двох чисел у полі Галуа *GF*(256). Для множення чисел, які розглядаються у даному прикладі, відповідно маємо.

1. Множення чисел 3 · 17.

Перша дія. За таблицею 3.15 знаходимо: $\log_2(3) = 25$; $\log_2(17) = 100$.

Друга дія. $\log_2(3) + \log_2(17) = 25 + 100 = 125$.

Третя дія. За таблицею 3.14 знаходимо: $2^{125} = 51$.

Тобто, остаточний результат для поля Галуа *GF*(256): $3 \cdot 17 = 51$.

2. Множення чисел 6 · 49.

Перша дія. За таблицею 3.15 знаходимо: $\log_2(6) = 181$; $\log_2(49) = 26$.

Друга дія. $\log_2(6) + \log_2(49) = 26 + 181 = 207$.

Третя дія. За таблицею 3.14 знаходимо: $2^{207} = 166$.

Тобто, остаточний результат для поля Галуа *GF*(256): $6 \cdot 49 = 166$.

3. Множення чисел $8 \cdot 49$.

Перша дія. За таблицею 3.15 знаходимо: $\log_2(8) = 3$; $\log_2(49) = 26$.

Друга дія. $\log_2(8) + \log_2(49) = 3 + 181 = 184$.

Третя дія. За таблицею 3.14 знаходимо: $2^{184} = 149$.

Тобто, остаточний результат для поля Галуа $GF(256)$: $8 \cdot 49 = 149$.

3. Множення чисел $25 \cdot 51$.

Перша дія. За таблицею 3.15 знаходимо: $\log_2(25) = 193$; $\log_2(51) = 125$.

Друга дія. $\log_2(25) + \log_2(51) = 193 + 125 = 318$; $318 > 255$; $318 - 255 = 63$.

Третя дія. За таблицею 3.14 знаходимо: $2^{63} = 161$.

Тобто, остаточний результат для поля Галуа $GF(256)$: $25 \cdot 51 = 161$.

Приклад 3.30. Скориставшись визначенням 3.11, формулою (3.12) та таблицями 3.14, 3.15 та 3.17, перевірити результати, отримані у прикладах 3.28 та 3.29, через операцію ділення чисел у полі Галуа $GF(2^m)$.

1. Перевіримо з використанням операції ділення чисел у полі Галуа $GF(16)$, що $3 \cdot 5 = 15$.

Таку перевірку можна зробити, впевнившись, що у полі Галуа $GF(16)$ $15/5 = 3$.

Звернемося до таблиці 3.17 та виконаємо наступні дії.

Перша дія. За таблицею 3.17 знаходимо: $\log_2(15) = 12$; $\log_2(5) = 8$.

Друга дія. $\log_2(15) - \log_2(5) = 12 - 8 = 4$.

Третя дія. За таблицею 3.17 знаходимо: $2^4 = 3$.

Тобто, $15/5 = 3$, звідки випливає, що $3 \cdot 5 = 15$.

2. Перевіримо з використанням операції ділення чисел у полі Галуа $GF(16)$, що $6 \cdot 7 = 1$.

Перевіримо, виконуючи операцію ділення, що тотожність $1/7 = 6$ для поля Галуа $GF(16)$ дійсно є вірною.

Згідно із таблицею 3.17 маємо.

Перша дія. $\log_2(1) = 0$; $\log_2(7) = 10$.

Друга дія. $\log_2(1) - \log_2(7) + 15 = 15 - 10 = 5$.

Третя дія. $2^5 = 6$.

3. Перевіримо з використанням операції ділення чисел у полі Галуа *GF*(16), що 12 · 13 = 3.

Тобто, необхідно перевірити для поля Галуа *GF*(16) коректність виразу 3/13 = 12.

Як і у попередніх випадках, скористаємось для цього таблицею 3.17.

Перша дія. $\log_2(3) = 4$; $\log_2(13) = 13$.

Друга дія. $\log_2(8) - \log_2(13) = 4 - 13 = -9$; $-9 < 0$; $-9 + 15 = 6$.

Третя дія. $2^6 = 12$.

Тобто, 3/13 = 12, звідки випливає, що 12 · 13 = 3.

4. Перевіримо з використанням операції ділення чисел у полі Галуа *GF*(16), що 11 · 14 = 8.

Для цього випадку необхідно перевірити для поля Галуа *GF*(16) коректність виразу 8/11 = 14.

За таблицею 3.17, після пошуку логарифмів діленого та дільника, їхнього віднімання та піднесення числа 2 до степені отриманого числа *i*, маємо наступні результати.

Перша дія. $\log_2(8) = 3$; $\log_2(11) = 7$.

Друга дія. $\log_2(8) - \log_2(11) = 3 - 7 = -4$; $-4 < 0$; $-4 + 15 = 11$.

Третя дія. $2^{11} = 14$.

Тобто, 8/11 = 14, звідки випливає, що 11 · 14 = 8.

5. Перевіримо з використанням операції ділення чисел у полі Галуа *GF*(16), що 14 · 14 = 11.

Для цього випадку необхідно впевнитися, що вірною є тотожність 11/14 = 14. З використанням даних, наведених у таблиці 3.17, отримуємо наступні результати.

Перша дія. $\log_2(11) = 7$; $\log_2(14) = 11$.

Друга дія. $\log_2(11) - \log_2(14) = -4$; $-4 < 0$; $-4 + 15 = 11$.

Третя дія. $2^{11} = 14$.

Тобто, 11/14 = 14, звідки випливає, що 14 · 14 = 11.

6. Перевіримо з використанням операції ділення чисел у полі Галуа *GF*(256), що $3 \cdot 17 = 51$.

У даному випадку необхідно показати, що у полі Галуа *GF*(256) виконується тотожність $51/17 = 3$.

Звертаючись до таблиць 3.14 та 3.15, для поля Галуа *GF*(256) отримуємо наступні результати.

Перша дія. $\log_2(51) = 125$; $\log_2(17) = 100$.

Друга дія. $\log_2(51) - \log_2(17) = 125 - 100 = 25$.

Третя дія. $2^{25} = 3$.

Тобто, у полі Галуа *GF*(256) $51/17 = 3$, звідки випливає, що $3 \cdot 17 = 51$.

7. Перевіримо з використанням операції ділення чисел у полі Галуа *GF*(256), що $6 \cdot 49 = 166$.

Для цього випадку

слід впевнитись у тому, що у полі Галуа *GF*(256) виконується тотожність $166/49 = 6$.

Згідно із таблицями 3.14 та 3.15, маємо наступні результати.

Перша дія. $\log_2(166) = 207$; $\log_2(49) = 181$.

Друга дія. $\log_2(166) - \log_2(49) = 207 - 181 = 26$.

Третя дія. $2^{26} = 6$.

Тобто, у полі Галуа *GF*(256) $166/49 = 6$, звідки випливає, що $6 \cdot 49 = 166$.

8. Перевіримо з використанням операції ділення чисел у полі Галуа *GF*(256), що $8 \cdot 49 = 149$.

У даному разі слід перевірити тотожність $149/49 = 8$.

Згідно із таблицями 3.14 та 3.15 можна записати.

Перша дія. $\log_2(149) = 184$; $\log_2(49) = 181$.

Друга дія. $\log_2(149) - \log_2(49) = 184 - 181 = 3$.

Третя дія. $2^3 = 8$.

Тобто, у полі Галуа *GF*(256) $149/49 = 8$, звідки випливає, що $8 \cdot 49 = 149$.

9. Перевіримо з використанням операції ділення чисел у полі Галуа *GF*(256), що $25 \cdot 51 = 161$.

У даному разі слід впевнитись у тому, що 161/25 = 51.

Згідно із таблицями 3.14 та 3.15, маємо наступні результати.

Перша дія. $\log_2(161) = 63$; $\log_2(25) = 193$.

Друга дія. $\log_2(161) - \log_2(25) = 63 - 193 = -130$; $-130 < 0$; $-130 + 255 = 125$.

Третя дія. $2^3 = 51$.

Тобто, у полі Галуа $GF(256)$ 161/25 = 51, звідки випливає, що $25 \cdot 51 = 161$.

Розглянувши більш досконало алгебраїчні операції у полі Галуа $GF(2^m)$, зокрема операції додавання, віднімання, множення, ділення, піднесення до степені та логарифму, можна тепер оцінити їх переваги та недоліки відносно операцій звичайної арифметики. З одного боку, дещо важко звикнути до того, що операції у обмеженому полі Галуа зазвичай дають дещо інший результат, ніж звичайні арифметичні. Дійсно, трішки незвично, що у полі Галуа $GF(16)$ $6 \cdot 7 = 1$, $2^5 = 6$, а $12 \cdot 13 = 3$.

Проте цей недолік пов'язаний лише із тим, що пересіченим людям, які звикли лише до операцій звичайної арифметики, важко зрозуміти, що таки звичайні операції, як додавання, множення та ділення, можуть у полі Галуа давати інший результат. Якщо зрозуміти та прийняти цю просту гіпотезу, в іншому алгебраїчні операції у полі Галуа $GF(2^m)$ не протерічать правилам арифметики. Зокрема, вони відповідають властивостям адитивності, комутативності та асоціативності, які були розглянуті у другій частині посібника. Для них існують зворотні операції, зокрема, ділення для операції множення та функція логарифму для степеневої функції.

Але функції множення, ділення та піднесення до степені у полях Галуа $GF(2^m)$ мають одну цікаву властивість, яка не притаманна звичайним арифметичним операціям. Річ у тому, що якщо звичайні арифметичні операції визначені на нескінченній множені дійсних чисел, операції у полі Галуа $GF(2^m)$ є зімкненими на елементах даного поля. Це легко зрозуміти для функції степені, аналізуючи формули (3.210) – (3.213), а також таблиці 3.14, 3.15, 3.16 та 3.17. Аналіз показує, що всі значення степеневої функції 2^i, для будь-якого поля Галуа порядку m, є зімкненими на елементах цього поля. Тобто, функція 2^i відображує елементи множини $\{0, 1, \ldots, 2^m - 1\}$ на елементи цієї самої

множини. І, що важливо, для мультиплікативних полів Галуа це відношення є гомоморфним, тобто, степенева функція 2^i є однозначною. А це дозволяє відразу визначити у полі Галуа $GF(2^m)$ зворотну функцію $\log_2(i)$, та через логарифми та степені, за правилами звичайної арифметики, виконати операції множення та ділення, які також є зімкненими на елементах даного поля.

Якщо, як було вказано вище, алгебраїчні операції у полі Галуа є вкрай непростими для розуміння пересіченими людьми із-за того, що вони не відповідають звичайним арифметичним операціям, комп'ютерна обробка числових даних з використанням операцій у полі Галуа $GF(2^m)$ значно спрощується. Це пояснюється тим, що зімкненість алгебраїчних операцій на елементах даного поля дозволяє запобігти виникнення помилкових ситуації, пов'язаних із переповненням розрядної сітки комп'ютера [13, 14, 31]. Насправді, для раціональних чисел закон комутативності для множення та ділення у комп'ютерній арифметиці не завжди виконуватися. Якщо a є занадто великим або занадто малим раціональним числом, а $b = 1/a$, у обчислювальній арифметиці значення $a \cdot b$ може суттєво відрізнятися від 1, і причиною цього завжди є значна втрата точності проведених обчислень.

Тому під час написання комп'ютерних програм з алгебраїчними операціями над полями Галуа $GF(2^m)$ можна не використовувати зайвих арифметичних перевірок, пов'язаних із аналізом можливості виникнення помилки через втрату точності. Це також є вагомою причиною для використання алгебраїчних операцій у полях Галуа для формування цифрових завадостійких кодів. Зокрема, на основі цих операцій побудовані коди БЧХ та Ріда – Соломона.

Єдиним недоліком комп'ютерної реалізації арифметики із полями Галуа є те, що якщо функція 2^i для будь-якого поля може бути легко обчисленою через ітераційне співвідношення (3.210), то для зворотної функції $\log_2(i)$ аналітичного виразу не існує. За таких умов функцію $\log_2(i)$ можна обчислити лише через аналіз значень функції 2^i та комбінаторне перебирання цих значень, а, як відомо, такий шлях потребує значного збільшення кількості елементарних операцій в обчислювальних алгоритмах [13, 14, 31]. А, згідно із

визначеннями 3.10 та 3.11, знання значень логарифмів у полі Галуа $GF(2^m)$ є необхідним для множення та ділення елементів цього поля. Тобто, у разі проведення складних обчислень, їх час значно збільшується через необхідність реалізації складних алгоритмів пошуку [48].

Коди комп'ютерних програм для обчислення функцій піднесення до степені, логарифму, множення, ділення та додавання для елементів полів Галуа $GF(2^m)$ наведені у додатку Н. У функції **`powgf2m(m,i)`** реалізований алгоритм розрахунку степеневої функції 2^i з використанням співвідношення (3.210), значення порядку поля **`m`** можуть бути або від 3 до 8, 12 або 16. Функція **`loggf2m(m,i)`** дозволяє обчислювати логарифми у визначеному полі Галуа порядку **`m`** через комбінаторне перебирання значень степеневої функції. Функція **`prodgf2m(m,i,j)`** призначена для обчислення добутку елементів поля Галуа $GF(2^m)$ **`i`** та **`j`** через значення їхніх логарифмів з використанням визначення 3.10, а функція **`divgf2m(m,i,j)`** – відповідно для ділення елементів **`i`** та **`j`** з використанням визначення 3.11. Для обчислення суми елементів поля Галуа порядку **`m`** через виконання логічної операції «виключного або» у функції **`sumgf2m(m,i,j)`** виконується операція перетворення десяткового числа до вектору його двійкового подання значеннями одиниць та нулів, а також зворотне перетворення вектора бітів до десяткового числа. Для виконання цих операцій були написані окремі функції. Функція **`conv2bin(i,m)`** призначена для перетворення десяткового числа до двійкового вектора з використанням алгоритму ділення на 2 та взяття остач, а функція **`convbin2dec(i,m)`** – для перетворення вектора двійкового подання числа до його десяткового формату з використанням алгоритму сумування вагових значень розрядів [13, 14, 31, 48].

Всі ці програми, наведені у додатку Н, є досить простими. Написані вони з використанням принципу модульного програмування [13, 14]. Єдиною суттєвою відмінною рисою цих програмних модулів є використання принципу матричного програмування для сумування векторів елементів у полях Галуа. Для цього виконуються відповідні матричні перетворення. Вектори елементів, які

сумуються, перетворюються до двійкових матриць, у стовпчиках яких записуються значення цих елементів у двійковому форматі. У такому запису кількість стовпчиків відповідає кількості елементів початкового вектора, а кількість рядків – порядку поля Галуа. Якщо два вектора однакової довжини, елементи яких належать полю Галуа одного порядку, перетворити до таких матриць, тоді результат поелементного сумування цих матриць за модулем два дасть двійкову матрицю, у стовпчиках якої будуть стояти вектори суми вхідних векторів за модулем два, записані у двійковому форматі. Для реалізації такої алгебраїчної операції засобами системи MatLab необхідно лише реалізувати вхідний контроль розмірності матриць. Кількість стовпчиків має відповідати розмірності векторів, а кількість рядків – порядку поля Галуа. Таке перетворення елементів поля Галуа пов'язано також із матричним поданням відношень між множинами, яке було розглянуто у підрозділі 4.3 другої частини посібника [48].

Для подання значень елементів поля Галуа через таку двійкову матрицю важливим є порядок слідування елементів двійкового запису числа. Оскільки перетворення векторів на двійкову матрицю є внутрішньою програмною процедурою, необхідно лише домовитись про відповідний порядок слідування розрядів та надалі дотримуватись його під час написання програми та формування алгоритмів обробки чисел. Наприклад, будемо вважати, що порядок є прямим, тобто, у верхніх рядках матриці розташовані старші розряди чисел, які сумуються, а у нижніх – молодші розряди. Саме з урахуванням такого розташування розрядів написані процедури перетворення чисел **`conv2bin(i,m)`** та **`convbin2dec(i,m)`**.

Розглянемо відповідний приклад.

Приклад 3.31. З використанням подання у вигляді двійкових матриць та матричних перетворень знайти суму векторів [3, 4, 5, 6] та [7, 5, 3, 1] у полі Галуа $GF(2^3)$.

Результат сумування цих векторів наочно показаний на рис. 3.39. Із наведеного прикладу зрозуміло, що елементи поля Галуа записуються та читаються у двійковій формі за стовпчиками матриць зверху до низу, що є

досить зручним під час ручного та комп'ютерного аналізу таких матриць.

$$[3,4,5,6]\rightarrow\begin{bmatrix}0&1&1&1\\1&0&0&1\\1&0&1&0\end{bmatrix}_{[3,4,5,6]}. \qquad [7,5,3,1]\rightarrow\begin{bmatrix}1&1&0&0\\1&0&1&1\\1&1&1&1\end{bmatrix}_{[7,5,3,1]}.$$

$$[3,4,5,6]\oplus[7,5,3,1]=\begin{bmatrix}0&1&1&1\\1&0&0&1\\1&0&1&0\end{bmatrix}\oplus\begin{bmatrix}1&1&0&0\\1&0&1&0\\1&1&1&1\end{bmatrix}=\begin{bmatrix}0\oplus1&1\oplus1&1\oplus0&1\oplus0\\1\oplus1&0\oplus0&0\oplus1&1\oplus0\\1\oplus1&0\oplus1&1\oplus1&0\oplus1\end{bmatrix}=$$

$$=\begin{bmatrix}1&0&1&1\\0&0&1&1\\0&1&0&1\end{bmatrix}_{[4,1,5,7]}=[4,1,5,7].$$

Рис. 3.29 Наочна ілюстрація сумування векторів [3, 4, 5, 6] та [7, 5, 3, 1] у полі Галуа $GF(2^3)$ з використанням двійкових матриць

Іншою особливістю наведених програмних модулів є реалізація функції множення та ділення елементів поля Галуа. Для програми множення вона полягає у тому, що першим множником може бути вектор, у той час як другий множник – це завжди число. Тобто функція множення елементів **`prodgf2m(m,i,j)`** має три параметри: **`m`** – порядок поля, а **`i`** – вектор елементів поля, який множиться на число **`j`** – елемент поля. Аналогічно у програмі ділення **`divgf2m(m,i,j)`** ділене може бути вектором, а дільник – це завжди число. Такий підхід є дуже зручним для реалізації поліноміальних операції у полях Галуа $GF(2^m)$, які розглядатимуться у наступному підрозділі.

Також є одна суттєва особливість комп'ютерної реалізації функції обчислення логарифму від елементів поля Галуа **`loggf2m(m,i)`**. Здається, що якщо існує програма для обчислення функція степені **`powgf2m(m,i)`**, можна реалізувати обчислення зворотної функції логарифму безпосередньо через виклик степеневої функції. Але тут проблема полягає у тому, що співвідношення (3.210) для обчислення степеневої функції є рекурентним, що збільшує необхідну кількість покрокових операцій, які потрібні для аналізу

значень цієї функції, від 2^m до (2^{m-1}!). Наприклад, якщо у полі Галуа $GF(2^m)$ обчислені значення функції 2^i для аргументу i від 0 до 99, а треба обчислити наступне значення для i = 100, тоді у разі виклику степеневої функції, яка працює за рекурентною за формулою (3.210), необхідно спочатку обчислити всі попередні значення. Проте, якщо відоме значення степеневої функції pow(99), нове значення pow(100) можна обчислити значно простіше, згідно із формулою (3.210), але лише за умови, що попереднє значення функції pow(99) було збережене. Необхідно помножити на 2 відому величину pow(99) та, у разі необхідності, додати до отриманого результату твірний поліном, записаний у числовій формі. Тому у програмному модулі **`loggf2m`** обчислення значення степеневої функції реалізовано безпосередньо через ітераційну формулу (3.210) та через використання попередніх обчислених значень pow(i – 1), як і у функції **`powgf2m`**. Такий підхід щодо проведення обчислень у полі Галуа є значно ефективнішим, особливо у разі необхідності виконання великої кількості операцій множення та ділення елементів поля, які, згідно із визначеннями 3.10 та 3.11, реалізуються через обчислення значень функцій логарифму та степені.

Також у додатку Н наведені результати тестування розглянутих програмних модулів. В подальшому ці програми будуть використані для реалізації поліноміальних операцій у полях Галуа $GF(2^m)$ та для побудови групових кодів Ріда – Соломона. У наступному підрозділі розглядатимуться поліноміальні операції у полях Галуа та особливості їхньої програмної реалізації.

3.4.4.2 Поліноміальні операції у полях Галуа $GF(2^m)$ та їхня програмна реалізація у системі MatLab

В цілому поліноміальні операції у полях Галуа $GF(2^m)$ базуються на елементарних операціях у таких полях, які були розглянуті у попередньому підрозділі. У загальному вигляді поліноміальні операції над елементами полів

Галуа були розглянуті у розділі 3 другої частини посібника [48], також вони досконало описані у навчальному посібнику [55].

Відомо, що поліном у полі Галуа записується як [55, 48, 81]:

$$A(x) = A_{k-1}x^{k-1} \oplus \ldots \oplus A_1 x + A_0 . \qquad (3.214)$$

З урахуванням співвідношення (3.211), суму двох поліномів у полі Галуа $GF(2^m)$ можна знайти наступним чином [81]:

$$A(x) \oplus B(x) = \sum_{i=0}^{k-1} (A_i \oplus B_i)x^i , \quad A_i, B_i \in GF\left(2^m\right), i = 0,1,\ldots,k-1 . \qquad (3.215)$$

Співвідношення (3.214) може бути реалізовано з використанням функції MatLab **`sumgf2m(m,i,j)`**, яка була розглянута у попередньому підрозділі. Як було відмічено вище, особливість реалізації цієї функції полягає у тому, що сумувати можна не лише числа, але і вектори, оскільки для сумування елементів використовуються функції обробки матриць двійкових відносин. Щодо операції віднімання поліномів, в алгебрі полів Галуа вона є еквівалентною операції додавання та замінюється логічною операцією «виключного або». Таким же чином у полях Галуа реалізована операція віднімання окремих елемкнтів [55, 81].

Іншою важливою поліноміальною операцією є множення та ділення поліному на число, яка у математичній формі може бути записана наступним чином [81]:

$$\lambda \cdot A(x) = \sum_{i=0}^{k-1} (\lambda \cdot A_i)x^i , \quad A_i, \lambda \in GF\left(2^m\right), i = 0,1,\ldots,k-1 . \qquad (3.216)$$

Множення поліному на число еквівалентно множенню вектора, який відповідає коефіцієнтам цього поліному, на це число, з урахуванням особливостей множення елементів полів Галуа $GF(2^m)$, які були розглянуті у попередньому підрозділі. А як було відмічено вище, операція множення вектора, створеного із елементів поля Галуа, на відповідний елемент цього поля, може бути здійснена з використанням написаної функції MatLab **`prodgf2m(m,i,j)`**. Приклади проведення таких обчислень з використанням цієї функції наведені у додатку Н.

Також важливою поліноміальною операцією в алгебрі полів Галуа є зсув коефіцієнтів поліному на одну позицію ліворуч, який відповідає множенню поліному на незалежну змінну *x*. Цю операцію у математичній формі можна записати наступним чином [81]:

$$x \cdot A(x) = \sum_{i=0}^{k-1} A_i \cdot x^{i+1} = \sum_{i=0}^{k-1} A_i^* \cdot x^i , \qquad (3.217)$$

$$A_{i+1}^* = A_i , A_0 = 0, i = 0,1,\ldots,k-1 .$$

Для реалізації поліноміальної операції зсуву коефіцієнтів з використанням засобів матричного програмування системи MatLab достатньо до вектору коефіцієнтів поліному дописати у молодший розряд число 0. Ця операція в системах матричного програмування є елементарною [13, 14, 48], тому тут окремо розглядати її не будемо.

Дещо складнішою є алгебраїчна операція множення двох поліномів, коефіцієнти яких належать полю Галуа $GF(2^m)$. В цілому цей процес множення відповідає алгебрі поліномів, розглянутій у третьому розділі другої частини посібника. Кожний коефіцієнт результуючого поліному обчислюється як сумування за степенями результатів множення одного поліному на коефіцієнти другого поліному. Тобто, перший поліном масштабується коефіцієнтами другого та здійснюється зсув отриманого результату ліворуч на кількість позицій, яка дорівнює степені змінної, на який множиться поліном. У математичній формі цю поліноміальну операцію можна записати наступним чином [81]:

$$A(x) \cdot B(x) = \sum_{i=0}^{k-1} A_i \cdot x^i \cdot \left(\sum_{j=0}^{l-1} B_j \cdot x^j \right) = \sum_{r=0}^{k+l-2} \left(\sum_{\substack{i=0 \; ij=0 \\ i+j=r}}^{k-1 \; l-1} A_i \cdot B_j \right) \cdot x^r . \qquad (3.218)$$

У підрозділі 2.5.8 було показано, що алгебраїчна операція множення двійкових поліномів є еквівалентною множенню двійкових чисел у стовпчик із зсувом розрядів ліворуч відповідно до їхніх вагових коефіцієнтів. Згідно із

співвідношенням (3.215) операція множення елементі полів Галуа $GF(2^m)$ також може бути реалізованою через множення полінома на число, зсув отриманого результату ліворуч, відповідно із ваговим коефіцієнтом розряду, та сумування отриманих результатів за модулем два. Задача також спрощується тим, що, як було відмічено у попередньому підрозділі, операція множення у полі Галуа $GF(2^m)$, на відміну від звичайної арифметики, є зімкненою на елементах поля. Це дозволяє уникнути необхідності урахування перенесення додаткових цифр, отриманих в результаті множення, в старші розряди цього результату. Таке перенесення також притаманне двійковій комп'ютерній арифметиці та зазвичай потребує контролю за можливістю виникнення обчислювальних помилок через переповнення розрядної сітки [31].

Розглянемо приклади множення двох поліномів у полі Галуа.

Приклад 3.32. У полі Галуа $GF(2^4)$ знайти добутки поліномів $\left(3\cdot x^4+0\cdot x^3+6\cdot x^2+10\cdot x+13\right)\cdot\left(5\cdot x^2+3\cdot x+12\right)$ та $\left(5\cdot x^3+14\cdot x+7\right)\cdot(3\cdot x+10)$.

Для розв'язування поставленої задачі скористаємося таблицею 3.17 та визначенням 3.10, яке описує у загальному вигляді алгебраїчну операцію множення чисел у полях Галуа.

$$\left(3\cdot x^4+0\cdot x^3+6\cdot x^2+10\cdot x+13\right)\cdot\left(5\cdot x^2+3\cdot x+12\right)=(3\cdot 5)\cdot x^6+(0\cdot 5)\cdot x^5+$$
$$+(6\cdot 5)\cdot x^4+(10\cdot 5)\cdot x^3+(13\cdot 5)\cdot x^2+(3\cdot 3)\cdot x^5+(0\cdot 3)\cdot x^4+(3\cdot 6)\cdot x^3+$$
$$+(10\cdot 3)\cdot x^2+(13\cdot 3)\cdot x+(12\cdot 3)\cdot x^4+(12\cdot 0)\cdot x^3+(12\cdot 6)\cdot x^2+(12\cdot 10)\cdot x+$$
$$+(12\cdot 13)=(3\cdot 5)\cdot x^6+((3\cdot 3)\oplus(0\cdot 5))\cdot x^5+((6\cdot 5)\oplus(12\cdot 3)\oplus(0\cdot 3))\cdot x^4+$$
$$+((10\cdot 5)\oplus(3\cdot 6)\oplus(12\cdot 0))\cdot x^3+((13\cdot 5)\oplus(10\cdot 3)\oplus(12\cdot 6))\cdot x^2+$$
$$+\left((13\cdot 3)\oplus(12\cdot 10)\right)\cdot x+(12\cdot 13).$$

Обчислимо коефіцієнти визначеного поліному для поля Галуа $GF(2^4)$, згідно із таблицею 3.17 та визначенням 3.10.

$$(3\cdot 5)=2^{(\log_2(3)+\log_2(5))}=2^{4+8}=2^{12}=15.$$

$$(3\cdot 3)\oplus(0\cdot 5)=\left(2^{(\log_2(3)+\log_2(3))}\right)\oplus 0=2^{4+4}=2^8=5$$

$$((6\cdot 5)\oplus(12\cdot 3)\oplus(0\cdot 3))=2^{(\log_2(6)+\log_2(5))}\oplus 2^{(\log_2(12)+\log_2(3))}\oplus 0=$$

$$= 2^{5+8} \oplus 2^{6+4} = 2^{13} \oplus 2^{10} = 13 \oplus 7 = 1101_2 \oplus 0111_2 = 1010_2 = 10_{10}.$$

$$((10 \cdot 5) \oplus (3 \cdot 6) \oplus (12 \cdot 0)) = 2^{(\log_2(10)+\log_2(5))} \oplus 2^{(\log_2(3)+\log_2(6))} =$$

$$= 2^{9+8-15} \oplus 2^{4+5} = 2^2 \oplus 2^9 = 4 \oplus 10 = 0100_2 \oplus 1010_2 = 1110_2 = 14_{10}.$$

$$((13 \cdot 5) \oplus (10 \cdot 3) \oplus (12 \cdot 6)) =$$

$$= \left(2^{(\log_2(13)+\log_2(5))} \oplus 2^{(\log_2(10)+\log_2(3))}\right) \oplus \left(2^{(\log_2(12)+\log_2(6))}\right) =$$

$$= \left(2^{13+8-15} \oplus 2^{9+4}\right) \oplus 2^{6+5} = \left(2^6 \oplus 2^{13}\right) \oplus 2^{11} = (12 \oplus 13) \oplus 14 =$$

$$= 1100_2 \oplus 1101_2 \oplus 1110_2 = 1111_2 = 15_{10}.$$

$$((13 \cdot 3) \oplus (12 \cdot 10)) = 2^{(\log_2(13)+\log_2(3))} \oplus 2^{(\log_2(12)+\log_2(10))} =$$

$$= 2^{13+4-15} \oplus 2^{6+9-15} = 2^2 \oplus 2^0 = 4 \oplus 1 = 0100_2 \oplus 0001_2 =$$

$$= 0101_2 = 1_{10}.$$

$$(12 \cdot 13) = 2^{(\log_2(12)+\log_2(13))} = 2^{6+13-15} = 2^4 = 3.$$

Остаточно маємо:

$$\left(3 \cdot x^4 + 0 \cdot x^3 + 6 \cdot x^2 + 10 \cdot x + 13\right) \cdot \left(5 \cdot x^2 + 3 \cdot x + 12\right) =$$

$$= 15 \cdot x^6 + 5 \cdot x^5 + 10 \cdot x^4 + 14 \cdot x^3 + 15 \cdot x^2 + 5 \cdot x + 3.$$

Для поліномів $\left(5 \cdot x^3 + 0 \cdot x^2 + 14 \cdot x + 7\right)$ та $(3 \cdot x + 10)$, відповідно, маємо:

$$\left(5 \cdot x^3 + 0 \cdot x^2 + 14 \cdot x + 7\right) \cdot (3 \cdot x + 10) = (5 \cdot 3) \cdot x^4 + (5 \cdot 10) \cdot x^3 + (0 \cdot 3) \cdot x^3 + (0 \cdot 10) \cdot x^2 +$$

$$+ (14 \cdot 3) \cdot x^2 + (14 \cdot 10) \cdot x + (7 \cdot 3) \cdot x + (7 \cdot 10) = (5 \cdot 3) \cdot x^4 +$$

$$+ ((5 \cdot 10) \oplus (0 \cdot 3)) \cdot x^3 + + ((14 \cdot 3) \oplus (0 \cdot 10)) \cdot x^2 +$$

$$+ ((14 \cdot 10) \oplus (7 \cdot 3)) \cdot x + (7 \cdot 10).$$

Коефіцієнти визначеного поліному четвертого порядку для поля Галуа *GF*(2^4) обчислюються наступним чином.

$$(5 \cdot 3) = 2^{\log_2(5)+\log_2(3)} = 2^{8+4} = 2^{12} = 15.$$

$$((5 \cdot 10) \oplus (0 \cdot 3)) = \left(2^{\log_2(5)+\log_2(10)}\right) \oplus 0 = \left(2^{8+9-15}\right) \oplus 0 = 2^2 = 4_{10}.$$

$$((14 \cdot 3) \oplus (0 \cdot 10)) = \left(2^{\log_2(14)+\log_2(3)}\right) \oplus 0 = 2^{11+4-15} = 2^0 = 1_{10}.$$

$$((14 \cdot 10) \oplus (7 \cdot 3)) = \left(2^{\log_2(14)+\log_2(10)}\right) \oplus \left(2^{\log_2(7)+\log_2(3)}\right) =$$

$$= \left(2^{11+9-15}\right) \oplus \left(2^{10+4}\right) = 2^5 \oplus 2^{14} = 6 \oplus 9 = 0110_2 \oplus 1001_2 =$$

$$= 1111_2 = 15_{10}.$$

$$(7 \cdot 10) = 2^{\log_2(7)+\log_2(10)} = 2^{10+9-15} = 2^4 = 3.$$

Остаточно маємо:

$$\left(5 \cdot x^3 + 14 \cdot x + 7\right) \cdot (3 \cdot x + 10) = 15 \cdot x^4 + 4 \cdot x^3 + x^2 + 15 \cdot x + 3.$$

Приклад 3.33. У полі Галуа $GF(2^8)$ знайти добуток поліномів $\left(135 \cdot x^5 + 47 \cdot x^4 + 0 \cdot x^3 + 112 \cdot x^2 + 134 \cdot x + 21\right) \cdot \left(115 \cdot x^3 + 0 \cdot x^2 + 225 \cdot x + 12\right)$.

Використовуючи таблиці 3.14 та 3.15 та визначення 3.10, отримуємо наступні результати.

$$\left(135 \cdot x^5 + 47 \cdot x^4 + 0 \cdot x^3 + 112 \cdot x^2 + 134 \cdot x + 21\right) \cdot \left(115 \cdot x^3 + 0 \cdot x^2 + 225 \cdot x + 12\right) =$$

$$= (135 \cdot 115) \cdot x^8 + (47 \cdot 115) \cdot x^7 + (0 \cdot 115) \cdot x^6 + (112 \cdot 115) \cdot x^5 +$$

$$+ (134 \cdot 115) \cdot x^4 + (21 \cdot 115) \cdot x^3 + (135 \cdot 0) \cdot x^7 + (47 \cdot 0) \cdot x^6 +$$

$$+ (0 \cdot 0) \cdot x^5 + (112 \cdot 0) \cdot x^4 + (134 \cdot 0) \cdot x^3 + (21 \cdot 0) \cdot x^2 + (135 \cdot 225) \cdot x^6 +$$

$$+ (47 \cdot 225) \cdot x^5 + (0 \cdot 225) \cdot x^4 + (112 \cdot 225) \cdot x^3 + (134 \cdot 225) \cdot x^2 +$$

$$+ (21 \cdot 225) \cdot x + (135 \cdot 12) \cdot x^5 + (47 \cdot 12) \cdot x^4 + (0 \cdot 12) \cdot x^3 + (112 \cdot 12) \cdot x^2 +$$

$$+ (134 \cdot 12)x + (21 \cdot 12) = (135 \cdot 115) \cdot x^8 + ((47 \cdot 115) \oplus (135 \cdot 0)) \cdot x^7 +$$

$$+ ((0 \cdot 115) \oplus (47 \cdot 0) \oplus (135 \cdot 225)) \cdot x^6 +$$

$$+ ((112 \cdot 115) \oplus (0 \cdot 0) \oplus (47 \cdot 225) \oplus (135 \cdot 12)) \cdot x^5 +$$

$$+ ((134 \cdot 115) \oplus (112 \cdot 0) \oplus (0 \cdot 225) \oplus (47 \cdot 12)) \cdot x^4 +$$

$$+ ((21 \cdot 115) \oplus (134 \cdot 0) \oplus (112 \cdot 225) \oplus (0 \cdot 12)) \cdot x^3 +$$

$$+ ((21 \cdot 0) \oplus (134 \cdot 225) \oplus (112 \cdot 12)) \cdot x^2 + ((21 \cdot 225) \oplus (134 \cdot 12)) \cdot x +$$

$$+ (21 \cdot 12).$$

$$(135 \cdot 115) = 2^{(\log_2(135)+\log_2(115))} = 2^{13+159} = 2^{172} = 123.$$

$$((47 \cdot 115) \oplus (135 \cdot 0)) = \left(2^{(\log_2(47)+\log_2(115))}\right) \oplus 0 =$$

$$= \left(2^{69+159}\right) \oplus 0 = 2^{228} \oplus 0 = 61_{10}.$$

$$((0 \cdot 115) \oplus (47 \cdot 0) \oplus (135 \cdot 225)) = 0 \oplus 0 \oplus \left(2^{(\log_2(135)+\log_2(225))}\right) =$$

$$= 0 \oplus 0 \oplus \left(2^{13+89}\right) = 0 \oplus 0 \oplus 68_{10} = 68_{10}.$$

$$((112 \cdot 115) \oplus (0 \cdot 0) \oplus (47 \cdot 225) \oplus (135 \cdot 12)) =$$

$$= ((112 \cdot 115) \oplus (47 \cdot 225) \oplus (135 \cdot 12)) = \left(2^{(\log_2(112)+\log_2(115))}\right) \oplus$$

$$\oplus \left(2^{(\log_2(47)+\log_2(225))}\right) \oplus \left(2^{(\log_2(135)+\log_2(12))}\right) =$$

$$= \left(2^{202+159-255}\right) \oplus \left(2^{69+89}\right) \oplus \left(2^{13+27}\right) = 2^{106} \oplus 2^{158} \oplus 2^{40} =$$

$$= 52_{10} \oplus 183_{10} \oplus 106_{10} = 00110100_2 \oplus 10110111_2 \oplus$$

$$\oplus 01101010_2 = 11101001_2 = 233_{10}.$$

$$((134 \cdot 115) \oplus (112 \cdot 0) \oplus (0 \cdot 225) \oplus (47 \cdot 12)) =$$

$$= (((134 \cdot 115) \oplus 0 \oplus 0 \oplus (47 \cdot 12)) =$$

$$\left(2^{(\log_2(134)+\log_2(115))}\right) \oplus \left(2^{(\log_2(47)+\log_2(12))}\right) =$$

$$= \left(2^{99+159-255}\right) \oplus \left(2^{69+27}\right) = 2^3 \oplus 2^{96} = 8_{10} \oplus 217_{10} =$$

$$= 00001000_2 \oplus 11011001_2 = 11010001_2 = 209_{10}.$$

$$((21 \cdot 115) \oplus (134 \cdot 0) \oplus (112 \cdot 225) \oplus (0 \cdot 12)) =$$

$$= ((21 \cdot 115) \oplus 0 \oplus (112 \cdot 225) \oplus 0) = \left(2^{(\log_2(21)+\log_2(115))}\right) \oplus$$

$$\oplus 0 \oplus \left(2^{(\log_2(112)+\log_2(225))}\right) \oplus 0 = \left(2^{141+159-255}\right) \oplus$$

$$\oplus \left(2^{202+89-255}\right) = 2^{45} \oplus 2^{36} = 193_{10} \oplus 37_{10} =$$

$$= 11000001_2 \oplus 00100101_2 = 11100100_2 = 228_{10}$$

$$(21 \cdot 0) \oplus (134 \cdot 225) \oplus (112 \cdot 12) = 0 \oplus (134 \cdot 225) \oplus (112 \cdot 12) =$$

$$= \left(2^{(\log_2(134)+\log_2(225))}\right) \oplus \left(2^{(\log_2(112)+\log_2(12))}\right) =$$

$$= 2^{99+89} \oplus 2^{202+27} = 2^{188} \oplus 2^{229} = 165_{10} \oplus 122_{10} =$$

$$= 10100101_2 \oplus 01111010_2 = 11011111_2 = 223_{10}.$$

$$((21 \cdot 225) \oplus (134 \cdot 12)) =$$

$$= \left(2^{(\log_2(21)+\log_2(225))}\right) \oplus \left(2^{(\log_2(134)+\log_2(12))}\right) =$$

$$= 2^{141+89} \oplus 2^{99+27} = 2^{230} \oplus 2^{126} = 244_{10} \oplus 102_{10} =$$

$$= 11110100_2 \oplus 01100110_2 = 10010010_2 = 146_{10}.$$

$$(21\cdot 12)=\left(2^{(\log_2(21)+\log_2(12))}\right)=2^{141+27}=2^{168}=252.$$

Остаточно маємо:

$$\left(135\cdot x^5+47\cdot x^4+0\cdot x^3+112\cdot x^2+134\cdot x+21\right)\times$$
$$\times\left(115\cdot x^3+0\cdot x^2+225\cdot x+12\right)=123\cdot x^8+61\cdot x^7+68\cdot x^6+$$
$$+233\cdot x^5+209\cdot x^4+228\cdot x^3+223\cdot x^2+146\cdot x+252.$$

Як і для множення звичайних та двійкових поліномів, пошук добутку поліномів у полі Галуа $GF(2^m)$ можна замінити на множення чисел стовпчиком. Таке множення здійснюється за правилами звичайної арифметики із додаванням нулів під час переходу до наступного розряду другого множника. Єдина різниця полягає у тому, що множення розрядів чисел здійснюється не за правилами звичайної арифметики, за правилами множення елементів полів Галуа, які були розглянуті у підрозділі 3.4.4.1.

Розглянемо приклади виконання поліноміальних операцій над елементами полів Галуа через множення векторів із елементами, які належать відповідному полю Галуа.

Приклад 3.34. Знайти добутки поліномів для прикладів 3.32 та 3.33 з використанням алгоритму множення векторів стовпчиком.

1. $\left(3\cdot x^4+0\cdot x^3+6\cdot x^2+10\cdot x+13\right)\cdot\left(5\cdot x^2+3\cdot x+12\right)$ для поля Галуа $GF(2^4)$.

$\left(3\cdot x^4+0\cdot x^3+6\cdot x^2+10\cdot x+13\right)\rightarrow[3,0,6,10,13]$.

$\left(5\cdot x^2+3\cdot x+12\right)\rightarrow[5,3,12]$.

$[3,0,6,10,13]\cdot 12=[3\cdot 12,\ 0\cdot 12,\ 6\cdot 12,\ 10\cdot 12,\ 13\cdot 12]=[7,\ 0,\ 14,\ 1,\ 3]$.

$[3,0,6,10,13]\cdot 3=[3\cdot 3,\ 0\cdot 3,\ 6\cdot 3,\ 10\cdot 3,\ 13\cdot 3]=[5,\ 0,\ 10,\ 13,\ 4]$.

$[3,0,6,10,13]\cdot 5=[3\cdot 5,\ 0\cdot 5,\ 6\cdot 5,\ 10\cdot 5,\ 13\cdot 5]=[15,\ 0,\ 13,\ 4,\ 12]$.

$$\begin{array}{r} 0\ \ 0\ \ 7\ \ 0\ 14\ 1\ 3 \\ 0\ \ 5\ \ 0\ 10\ 13\ 4\ 0 \\ \oplus\ 15\ 0\ 13\ \ 4\ \ 12\ 0\ 0 \\ \hline 15\ 5\ 10\ 14\ 15\ 5\ 3 \end{array}$$

Рис. 3.30 Ілюстрація множення векторів $[3,0,6,10,13]$ та $[5,3,12]$ у полі Галуа $GF(2^4)$ через матричні перетворення

Дійсно: $0 \oplus 0 \oplus 15 = 15$; $0 \oplus 5 \oplus 0 = 5$; $7 \oplus 0 \oplus 13 = 10$; $0 \oplus 10 \oplus 4 = 14$; $14 \oplus 13 \oplus 12 = 15$; $1 \oplus 4 \oplus 0 = 5$; $3 \oplus 0 \oplus 0 = 3$.

2. $\left(5 \cdot x^3 + 0 \cdot x^2 + 14 \cdot x + 7\right) \cdot \left(3 \cdot x + 10\right)$ для поля Галуа $GF(2^4)$.

$\left(5 \cdot x^3 + 0 \cdot x^2 + 14 \cdot x + 7\right) \rightarrow [5,\ 0,\ 14,\ 7]$. $\quad \left(3 \cdot x + 10\right) \rightarrow [3,\ 10]$.

$$[5,\ 0,\ 14,\ 7] \cdot 10 = [4,\ 0,\ 6,\ 3]. \quad [5,\ 0,\ 14,\ 7] \cdot 3 = [15,\ 0,\ 1,\ 9].$$

$$\begin{array}{r} 0\ \ 4\ \ 0\ \ 6\ \ 3 \\ \oplus 15\ \ 0\ \ 1\ \ 9\ \ 0 \\ \hline 15\ \ 4\ \ 1\ 15\ \ 3 \end{array}$$

Рис. 3.31 Ілюстрація множення векторів $[5,\ 0,\ 14,\ 7]$ та $[3,\ 10]$ у полі Галуа $GF(2^4)$ через матричні перетворення

Дійсно: $0 \oplus 15 = 15$; $4 \oplus 0 = 4$; $0 \oplus 1 = 1$; $6 \oplus 9 = 15$; $3 \oplus 0 = 3$.

3. $\left(135 \cdot x^5 + 47 \cdot x^4 + 0 \cdot x^3 + 112 \cdot x^2 + 134 \cdot x + 21\right) \cdot \left(115 \cdot x^3 + 0 \cdot x^2 + 225 \cdot x + 12\right)$ для поля Галуа $GF(2^8)$.

$\left(135 \cdot x^5 + 47 \cdot x^4 + 0 \cdot x^3 + 112 \cdot x^2 + 134 \cdot x + 21\right) \rightarrow [135,\ 47,\ 0,\ 112,\ 134,\ 21]$.

$\left(115 \cdot x^3 + 0 \cdot x^2 + 225 \cdot x + 12\right) \rightarrow [115,\ 0,\ 225,\ 12]$.

$$[135,\ 47,\ 0,\ 112,\ 134,\ 21] \cdot 12 = [106,\ 217,\ 0,\ 122,\ 102,\ 252].$$

$$[135,\ 47,\ 0,\ 112,\ 134,\ 21] \cdot 255 = [68,\ 183,\ 0,\ 37,\ 165,\ 244].$$

$$[135,\ 47,\ 0,\ 112,\ 134,\ 21] \cdot 0 = [0,\ 0,\ 0,\ 0,\ 0,\ 0].$$

$$[135,\ 47,\ 0,\ 112,\ 134,\ 21] \cdot 115 = [123,\ 61,\ 0,\ 52,\ 8,\ 193].$$

$$\begin{array}{r} 0\ \ 0\ \ 0\ 106\ 217\ \ 0\ 122\ 102\ 252 \\ 0\ \ 0\ 68\ 183\ \ 0\ \ 37\ 165\ 244\ \ 0 \\ \oplus\ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0 \\ 123\ 61\ \ 0\ \ 52\ \ 8\ 193\ \ 0\ \ 0\ \ 0 \\ \hline 123\ 61\ 68\ 233\ 209\ 228\ 223\ 146\ 252 \end{array}$$

Рис. 3.32 Ілюстрація множення векторів $[135,\ 47,\ 0,\ 122,\ 134,\ 21]$ та $[115,\ 0,\ 225,\ 12]$ у полі Галуа $GF(2^8)$ через матричні перетворення

Дійсно:

$0 \oplus 0 \oplus 0 \oplus 123 = 123$; $0 \oplus 0 \oplus 0 \oplus 61 = 61$; $0 \oplus 68 \oplus 0 \oplus 0 = 68$;

$106 \oplus 183 \oplus 0 \oplus 52 = 233$; $217 \oplus 0 \oplus 0 \oplus 0 = 217$; $0 \oplus 37 \oplus 0 \oplus 193 = 228$;

$122 \oplus 165 \oplus 0 \oplus 0 = 233$; $102 \oplus 244 \oplus 0 \oplus 0 = 146$; $252 \oplus 0 \oplus 0 \oplus 0 = 252$.

Видно, що отримані результати повністю співпадають із результатами множення поліномів за правилами поліноміальної алгебри, отриманими в прикладах 3.32 та 3.33.

Код програми **`prodpolgf2m`**, призначеної для обчислення добутку двох поліномів, коефіцієнтами яких є елементи поля Галуа, а також результати тестування цієї програми, наведені у додатку Н. У програмі реалізований матричний алгоритм множення векторів стовпчиком із зсувом розрядів вліво та сумуванням отриманих результатів множення за модулем два. Такий спосіб множення поліномів над елементами полів Галуа був досконало розглянутий у прикладі 3.34. Під час написання програми **`prodpolgf2m`** були реалізовані загальновідомі засоби матричного програмування системи MatLab [13, 14].

Слід відзначити, що, з одного боку, множення чисел, записаних стовпчиком, через операції у полі Галуа є дещо незвичним, оскільки постійно приходиться звертатися до таблиць 3.14 – 3.17, де наведені результати піднесення до степені та взяття логарифму для елементів полів Галуа $GF(2^3)$, $GF(2^4)$ та $GF(2^8)$. Проте ці результати для операції піднесення до степені можна отримати із рекурентного співвідношення (3.210), тоді операція взяття логарифму може бути виконана через перебирання отриманих значень степеневої функції. З іншого боку, суттєвою перевагою алгебраїчних операцій у полях Галуа є їх зімкненість на елементах поля, що дозволяє уникнути під час сумування результатів множення перенесень до сусіднього лівого розряду. Це було наочно видно під час розгляду прикладу 3.34. Під час написання програми **`prodpolgf2m`** були використані інші програмні модулі, призначені для здійснення алгебраїчних операцій над елементами полів Галуа, які розглядалися у підрозділі 3.3.4.1. Тобто, програма **`prodpolgf2m`** написана з використанням засобів матричного та модульного програмування системи

науково-технічних розрахунків MatLab [13, 14].

Аналогічно побудована програма **`divpolgf2m`** для ділення поліномів стовпчиком, яке виконується за схемою Горнера, що була описана у третьому розділі другої частини посібника. Єдина особливість цієї програми полягає у тому, що ділення здійснюється лише на приведений поліном, у якому коефіцієнт перед вищою степінню дорівнює 1, у противному випадку проводиться скорочення коефіцієнтів поліному шляхом ділення на старший коефіцієнт. Як буде показано у наступному підрозділі, для формування кодів Ріда – Соломона використовуються лише приведені поліноми.

Розглянемо приклад множення поліному на приведений поліном у полі Галуа $GF(2^4)$ та перевіримо результат множення через операцію ділення.

Приклад 3.35. Знайти добуток поліномів $\left(x^3+5\cdot x^2+12\cdot x+7\right)$ та $\left(x^2+10\cdot x+6\right)$ у полі Галуа $GF(2^4)$ та перевірити результат з використанням операції ділення поліномів за схемою Горнера.

Помножимо поліноми за алгоритмом множення векторів стовпчиком, який був розглянутий у прикладі 3.34.

$\left(x^3+5\cdot x^2+12\cdot x+7\right)\to[1,5,12,7]. \quad \left(x^2+10\cdot x+6\right)\to[1,10,6].$

$[1,5,12,7]\cdot 6=[1\cdot 6,\ 5\cdot 6,\ 12\cdot 6,\ 7\cdot 6]=[6,\ 13,\ 14,\ 1].$

$[1,5,12,7]\cdot 10=[1\cdot 10,\ 5\cdot 10,\ 12\cdot 10,\ 7\cdot 10]=[10,\ 4,\ 1,\ 3].$

$$\begin{array}{r} 0\ \ 0\ \ 6\ \ 13\ 14\ 1 \\ \oplus\ 0\ 10\ \ 4\ \ 1\ \ 3\ \ 0 \\ \hline 1\ \ 5\ \ 12\ \ 7\ \ 0\ \ 0 \\ \hline 1\ 15\ 14\ 11\ 13\ 1 \end{array}$$

Рис. 3.33 Ілюстрація множення векторів $[1,5,12,7]$ та $[1,10,6]$ у полі Галуа $GF(2^4)$ через матричні перетворення

Тобто, $\left(x^3+5\cdot x^2+12\cdot x+7\right)\cdot\left(x^2+10\cdot x+6\right)=$

$$=x^5+15\cdot x^4+14\cdot x^3+11\cdot x^2+13\cdot x+1.$$

Перевіримо отриманий результат через ділення поліномів за схемою Горнера.

$$
\begin{array}{l}
\oplus\ x^5 + 15\cdot x^4 + 14\cdot x^3 + 11\cdot x^2 + 13\cdot x + 1 \\
\quad\ x^5 + 10\cdot x^4 + 6\cdot x^3 \\
\hline
\quad\ 5\cdot x^4 + 8\cdot x^3 + 11\cdot x^2 \\
\quad \oplus\ 5\cdot x^4 + 4\cdot x^3 + 13\cdot x^2 \\
\hline
\quad\ 12\cdot x^3 + 6\cdot x^2 + 13\cdot x \\
\quad \oplus\ 12\cdot x^3 + 1\cdot x^2 + 14\cdot x \\
\hline
\quad\ 7\cdot x^2 + 3\cdot x + 1 \\
\quad \oplus\ 7\cdot x^2 + 3\cdot x + 1 \\
\hline
\quad\ 0
\end{array}
\quad\Bigg|\quad
\begin{array}{l}
x^2 + 10\cdot x + 6 \\
\hline
x^3 + 5\cdot x^2 + 12\cdot x + 7
\end{array}
$$

Рис. 3.34 Ілюстрація ділення поліному $x^5 + 15\cdot x^4 + 14\cdot x^3 + 11\cdot x^2 + 13\cdot x + 1$ на поліном $x^2 + 10\cdot x + 6$ у полі Галуа $GF(2^4)$ за схемою Горнера

Дійсно, у полі Галуа $GF(2^4)$ маємо:

$15 \oplus 10 = 5$; $14 \oplus 6 = 8$; $[1,10,6]\cdot 5 = [5,4,13]$.

$8 \oplus 4 = 12$; $11 \oplus 13 = 6$; $[1,10,6]\cdot 12 = [12,1,14]$.

$13 \oplus 14 = 3$; $[1,10,6]\cdot 7 = [7,3,1]$.

Приклад 3.36. До результату множення поліномів $\left(x^3 + 5\cdot x^2 + 12\cdot x + 7\right)$ та $\left(x^2 + 10\cdot x + 6\right)$ у полі Галуа $GF(2^4)$, отриманого у прикладі 3.35, додати поліном $(3\cdot x + 12)$, та знайти остачу від ділення цього поліному на поліном $\left(x^2 + 10\cdot x + 6\right)$.

$$\left(x^5 + 15\cdot x^4 + 14\cdot x^3 + 11\cdot x^2 + 13\cdot x + 1\right) \oplus (3\cdot x + 12) =$$

$$= x^5 + 15\cdot x^4 + 14\cdot x^3 + 11\cdot x^2 + (13 \oplus 3)\cdot x + (12 \oplus 1) =$$

$$= x^5 + 15\cdot x^4 + 14\cdot x^3 + 11\cdot x^2 + 14\cdot x + 13.$$

Ітераційний процес ділення отриманого поліному на поліном $x^2 + 10\cdot x + 6$ за схемою Горнера показаний на рис. 3.35. Цілком зрозуміло, що остача від такого ділення дорівнює $3\cdot x + 12$.

$$
\begin{array}{l|l}
\oplus\ x^5 + 15\cdot x^4 + 14\cdot x^3 + 11\cdot x^2 + 14\cdot x + 13 & x^2 + 10\cdot x + 6 \\
\quad\ x^5 + 10\cdot x^4 + 6\cdot x^3 & \hline
\quad\ x^3 + 5\cdot x^2 + 12\cdot x + 7 \\
\hline
\quad\oplus\ 5\cdot x^4 + 8\cdot x^3 + 11\cdot x^2 & \\
\quad\ \ 5\cdot x^4 + 4\cdot x^3 + 13\cdot x^2 & \\
\hline
\quad\quad\oplus\ 12\cdot x^3 + 6\cdot x^2 + 14\cdot x & \\
\quad\quad\ \ 12\cdot x^3 + 1\cdot x^2 + 14\cdot x & \\
\hline
\quad\quad\quad\oplus\ 7\cdot x^2 + 0\cdot x + 13 & \\
\quad\quad\quad\ \ 7\cdot x^2 + 3\cdot x + 1 & \\
\hline
\quad\quad\quad\quad 3\cdot x + 12 &
\end{array}
$$

Рис. 3.35 Ілюстрація ділення поліному $x^5 + 15\cdot x^4 + 14\cdot x^3 + 11\cdot x^2 + 14\cdot x + 13$ на поліном $x^2 + 10\cdot x + 6$ у полі Галуа $GF(2^4)$ за схемою Горнера

Алгоритм ділення поліномів у полі Галуа $GF(2^4)$ за схемою Горнера реалізований у програмі **divpolgf2m**, наведений у додатку Н. Результатом роботи цієї програми є матриця розмірністю (2, *n*), перший рядок якої містить частку від ділення, а другий – остачу. В цілому ця програма написана аналогічно програмі для ділення двійкових поліномів **Bin_Division**, яка була наведена у додатку К. Різниця полягає у тому, що множення поліномів над елементами поля Галуа на число здійснюється з використанням функції **prodgf2m**, яка була розглянута у попередньому підрозділі та реалізує алгебраїчну операцію множення вектора на число у полях Галуа.

Алгебраїчні засоби роботи з поліномами є більш зручними для проведення ручних розрахунків, проте для комп'ютерної реалізації більш ефективними є алгоритми множення та ділення чисел стовпчиком, основані на обробці векторів. У будь-якому разі, ці подання поліноміальних оппеарцій є еквівалентними. Алгоритми роботи з поліномами у полях Галуа через векторне подання поліноміальних коефіцієнтів реалізовані у програмах **prodgf2m** та **divgf2m**, написаних мовою програмування системи MatLab та наведених у додатк Н. Ці програми засобі використані у програмі, призначеній для формування кодів Ріда – Соломона та декодування їхніх послідовностей, яка наведена у додатку О.

3.4.5 Алгоритм формування кодів Ріда – Соломона та відповідні приклади

Спосіб формування кодів Ріда – Соломона є єдиним і досить простим. Загалом він базується на тому, що код Ріда – Соломона – це циклічний код, алгебраїчні операції в якому виконуються у полі Галуа $GF(2^m)$.

Повернемося до визначення 3.2 та формули (3.200). Згідно із цим співвідношенням код Ріда – Соломона формується у полі Галуа $GF(2^m)$ як систематичний циклічний код із твірним поліномом $g(x)=\prod_{i=1}^{2\cdot t}\left(x-2^i\right)$, де t – кількість помилок, які виправляються. Сутність цього співвідношення полягає у тому, що за умови кратності помилок t відповідна кількість розрядів коду t призначена для формування локатору помилки, тобто визначення помилкових позицій, а інші t розрядів – для формування вірних значень помилкових символів.

Наведемо приклади обчислення твірного поліному для РС – кодів.

Приклад 3.37. Обчислити твірний поліном $g(x)$ за умови $t = 3$ у полях Галуа $GF(2^3)$, $GF(2^4)$, $GF(2^5)$ та $GF(2^6)$.

Для поля $GF(2^3)$:

$$g(x)=\prod_{i=1}^{6}\left(x-2^i\right)=\left(x-2^1\right)\cdot\left(x-2^2\right)\cdot\left(x-2^3\right)\times$$
$$\times\left(x-2^4\right)\cdot\left(x-2^5\right)\cdot\left(x-2^6\right)=(x-2)\cdot(x-4)\cdot(x-3)\cdot(x-6)\times$$
$$\times(x-7)\cdot(x-5)=\left(x^2+6\cdot x+3\right)\cdot(x-3)\cdot(x-6)\cdot(x-7)\cdot(x-5)=$$
$$=\left(x^3+5\cdot x^2+2\cdot x+5\right)\cdot(x-6)\cdot(x-7)\cdot(x-5)=$$
$$=\left(x^4+3\cdot x^3+1\cdot x^2+2\cdot x+3\right)\cdot(x-7)\cdot(x-5)=$$
$$=\left(x^5+4\cdot x^4+3\cdot x^3+5\cdot x^2+6\cdot x+2\right)\cdot(x-5)=$$
$$=x^6+x^5+x^4+x^3+x^2+x+1.$$

Для поля $GF(2^4)$:

$$g(x)=\prod_{i=1}^{6}\left(x-2^i\right)=\left(x-2^1\right)\cdot\left(x-2^2\right)\cdot\left(x-2^3\right)\times$$

$$\times\left(x-2^4\right)\cdot\left(x-2^5\right)\cdot\left(x-2^6\right)=(x-2)\cdot(x-4)\cdot(x-8)\cdot(x-3)\times$$
$$\times(x-6)\cdot(x-12)=\left(x^2+6\cdot x+8\right)\cdot(x-8)\cdot(x-3)\times$$
$$\times(x-6)\cdot(x-12)=(x-6)\cdot(x-12)=$$
$$=\left(x^3+14\cdot x^2+13\cdot x+12\right)\cdot(x-3)\cdot(x-6)\cdot(x-12)=$$
$$=\left(x^4+13\cdot x^3+12\cdot x^2+8\cdot x+7\right)\cdot(x-6)\cdot(x-12)=$$
$$=\left(x^5+11\cdot x^4+4\cdot x^3+6\cdot x^2+2\cdot x+1\right)\cdot(x-12)=$$
$$=x^6+7\cdot x^5+9\cdot x^4+3\cdot x^3+12\cdot x^2+10\cdot x+12.$$

Для поля $GF(2^5)$:

$$g(x)=\prod_{i=1}^{6}\left(x-2^i\right)=\left(x-2^1\right)\cdot\left(x-2^2\right)\cdot\left(x-2^3\right)\times$$
$$\times\left(x-2^4\right)\cdot\left(x-2^5\right)\cdot\left(x-2^6\right)=(x-2)\cdot(x-4)\cdot(x-8)\cdot(x-16)\times$$
$$\times(x-5)\cdot(x-10)=\left(x^2+6\cdot x+8\right)\cdot(x-8)\cdot(x-16)\cdot(x-5)\cdot(x-10)=$$
$$=\left(x^3+14\cdot x^2+29\cdot x+10\right)\cdot(x-16)\cdot(x-5)\cdot(x-10)=$$
$$=\left(x^4+30\cdot x^3+6\cdot x^2+9\cdot x+17\right)\cdot(x-5)\cdot(x-10)=$$
$$=\left(x^5+27\cdot x^4+15\cdot x^3+23\cdot x^2+25\cdot x+31\right)\cdot(x-10)=$$
$$=x^6+17\cdot x^5+26\cdot x^4+30\cdot x^3+27\cdot x^2+30\cdot x+24.$$

Для поля $GF(2^6)$:

$$g(x)=\prod_{i=1}^{6}\left(x-2^i\right)=\left(x-2^1\right)\cdot\left(x-2^2\right)\cdot\left(x-2^3\right)\times$$
$$\times\left(x-2^4\right)\cdot\left(x-2^5\right)\cdot\left(x-2^6\right)=(x-2)\cdot(x-4)\cdot(x-8)\cdot(x-16)\times$$
$$\times(x-32)\cdot(x-3)=\left(x^2+6\cdot x+8\right)\cdot(x-8)\cdot(x-16)\cdot(x-32)\cdot(x-3)=$$
$$=\left(x^3+14\cdot x^2+56\cdot x+3\right)\cdot(x-16)\cdot(x-32)\cdot(x-3)=$$
$$=\left(x^4+30\cdot x^3+29\cdot x^2+17\cdot x+48\right)\cdot(x-32)\cdot(x-3)=$$
$$=\left(x^5+62\cdot x^4+12\cdot x^3+35\cdot x^2+8\cdot x+40\right)\cdot(x-3)=$$
$$=x^6+61\cdot x^5+13\cdot x^4+55\cdot x^3+46\cdot x^2+48\cdot x+59.$$

Слід зазначити, що операції множення поліномів у полях Галуа $GF(2^3)$ – $GF(2^6)$ у наведеному прикладі 3.37 описані не повністю. Це пов’язано із двома

причинами. По-перше, поліноміальні операції над елементами полів Галуа вже були досконало розглянуті у прикладах 3.32 та 3.33, а по-друге, для перевірки коректності виконання операції множення поліномів можна використовувати комп'ютерну програму **`prodpolgf2m`**, наведену у додатку Н.

Надалі, за умови відомих твірних поліномів, коди Ріда – Соломона формуються за звичайним правилом, як будь-які систематичні циклічні коди. Тобто, алгоритм формування кодів Ріда – Соломона є наступним.

1. З використанням співвідношення (3.200) за умови відомої кількості помилок t, які необхідно виправляти, формується твірний поліном. Спосіб формування твірних поліномів для різних значень степені поля m був розглянутий у прикладі 3.37.

2. До молодших розрядів вхідної послідовності, яку необхідно закодувати, додається послідовність нулів, кількість яких становить r , де r – степінь твірного поліному. Таким чином, на основі вхідного слова, створюється модифіковане вхідне слово.

3. Модифіковане вхідне слово ділиться на твірний поліном та визначається остача.

4. Визначена остача додається за правилами алгебри полів Галуа, тобто за модулем два, до модифікованого вхідного слова.

З точки зору математичного формалізму цей алгоритм можна записати у наступному вигляді:

$$r = 2 \cdot t; \qquad g(x) = \prod_{i=1}^{r} \left(x \oplus 2^i\right); \; M_m(x) = x^r M(x); \tag{3.219}$$

$$R(x) = \left(M_m(x)\right) \bmod \left(g(x)\right); \qquad F(x) = M(x) \oplus R(x),$$

де $g(x)$ – твірний поліном, $M(x)$ – вхідне слово, яке кодується, $M_m(x)$ – модифіковане вхідне слово, $R(x)$ – остача від ділення модифікованого вхідного слова на твірний поліном, $F(x)$ – результат кодування. Блок-схема алгоритму формування кодів Ріда – Соломона у полях Галуа $GF(2^m)$ наведена на рис. 3.36.

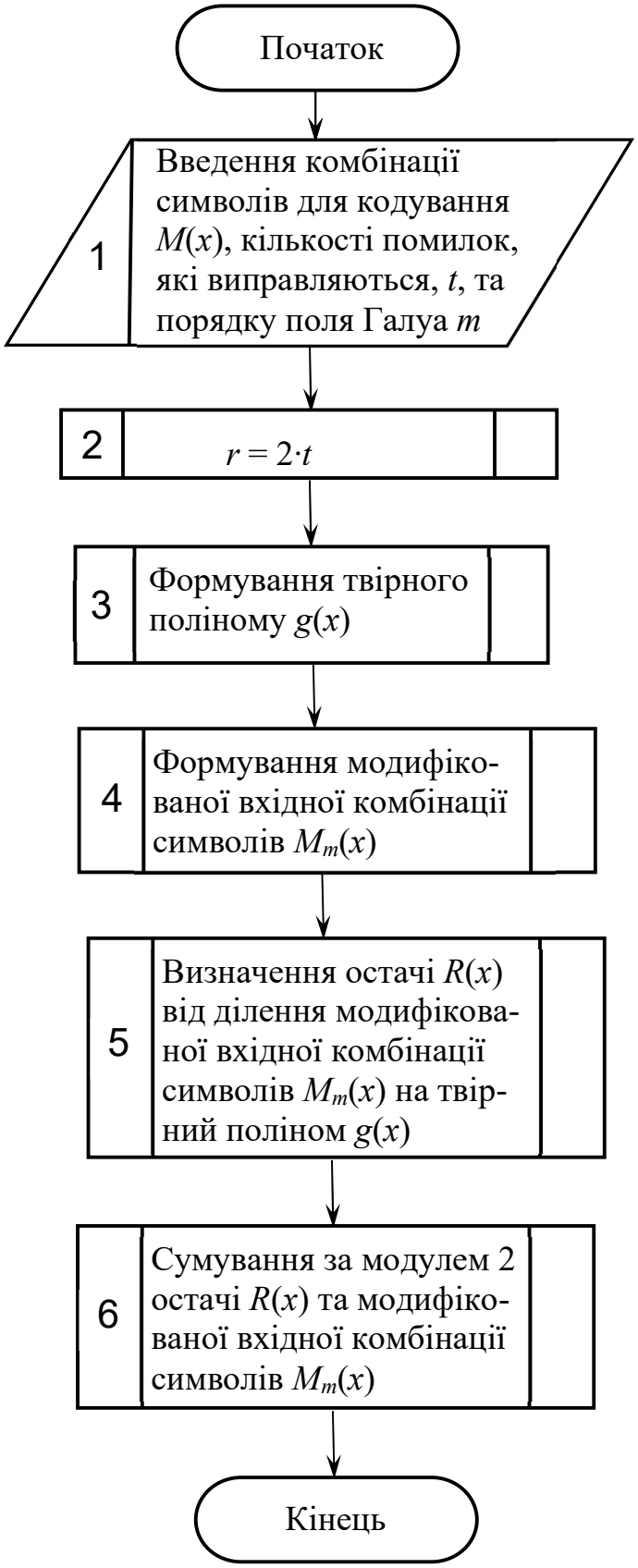


Рис. 3.36 Алгоритм формування кодів Ріда – Соломона

Приклад 3.38. Побудувати систематичний код Ріда – Соломона для кодової послідовності [14, 6, 13, 7] у полі Галуа $GF(2^5)$, вважаючи, що кількість

помилок, які необхідно виправляти, становить $t = 3$.

Твірний поліном для такого коду був обчислений у прикладі 3.37 та становить $x^6 + 17 \cdot x^5 + 26 \cdot x^4 + 30 \cdot x^3 + 27 \cdot x^2 + 30 \cdot x + 24$. Це поліном шостого порядку. Тому для формування коду Ріда – Соломона для визначеної вхідної кодової послідовності необхідно додати шість нулів до молодших розрядів та поділити модифіковану вхідну послідовність на твірний поліном. Процес ділення цих поліномів стовпчиком за схемою Горнера наочно показаний на рис. 3.37. Видно, що остача від ділення становить $29 \cdot x^5 + 29 \cdot x^4 + 27 \cdot x^3 + 21 \cdot x^2 + 9 \cdot x + 16$. За таких умов код Ріда –Соломона записується у вигляді:

$$RS = 14 \cdot x^9 + 6 \cdot x^8 + 13 \cdot x^7 + 7 \cdot x^6 + 29 \cdot x^5 + 29 \cdot x^4 + 27 \cdot x^3 + 21 \cdot x^2 + 9 \cdot x + 16 =$$
$$= [14, 6, 13, 7, 29, 29, 27, 21, 9, 16].$$

Оскільки до діленого додана остача, зрозуміло, що за такої умови остача від ділення отриманої кодової послідовності на твірний поліном у полі Галуа $GF(2^5)$ дорівнює нулю. Це цілком відповідає теоретичним відомостям про систематичні коди, наведеним у підрозділі 2.2.5 цієї частини посібника.

Приклад 3.39. Побудувати систематичний код Ріда – Соломона для кодової послідовності [46, 53, 39, 27] у полі Галуа $GF(2^6)$, вважаючи, що кількість помилок, які необхідно виправляти, становить $t = 3$.

Для поля Галуа $GF(2^6)$ твірний поліном також був отриманий у прикладі 3.37 та становить $x^6 + 61 \cdot x^5 + 13 \cdot x^4 + 55 \cdot x^3 + 46 \cdot x^2 + 48 \cdot x + 59$. Як і в попередньому прикладі, для формування коду Ріда – Соломона необхідно до вхідної кодової послідовності додати шість нулів до молодших розрядів та поділити модифіковану вхідну послідовність на твірний поліном. Зрозуміло, що степінь твірного поліному та кількість нулів, які дописуються до вхідної послідовності, не залежать від порядку поля Галуа $GF(2^m)$, на відміну від остаточного результату кодування. Процес ділення модифікованої кодової послідовності на твірний поліном наочно показаний на рис. 3.38. У цьому разі остача від ділення складає $21 \cdot x^5 + 48 \cdot x^4 + 38 \cdot x^3 + 24 \cdot x^2 + 36 \cdot x + 53$, тому код Ріда – Соломона записується у вигляді:

$$RS = 46 \cdot x^9 + 53 \cdot x^8 + 39 \cdot x^7 + 27 \cdot x^6 + 21 \cdot x^5 +$$
$$+ 48 \cdot x^4 + 38 \cdot x^3 + 24 \cdot x^2 + 36 \cdot x + 53 = [46, 53, 39, 27, 21, 48, 38, 24, 36, 53].$$

$$
\begin{array}{l|l}
\oplus\ \begin{array}{l} 14\cdot x^9 + 6\cdot x^8 + 13\cdot x^7 + 7\cdot x^6 + 0\cdot x^5 + 0\cdot x^4 + 0\cdot x^3 + 0\cdot x^2 + 0\cdot x + 0 \\ 14\cdot x^9 + 21\cdot x^8 + 24\cdot x^7 + 5\cdot x^6 + 22\cdot x^5 + 5\cdot x^4 + 4\cdot x^3 \end{array} & x^6 + 17\cdot x^5 + 26\cdot x^4 + 30\cdot x^3 + 27\cdot x^2 + 31\cdot x + 24 \\
\hline
\oplus\ \begin{array}{l} 19\cdot x^8 + 21\cdot x^7 + 2\cdot x^6 + 22\cdot x^5 + 5\cdot x^4 + 4\cdot x^3 + 0\cdot x^2 \\ 19\cdot x^8 + 11\cdot x^7 + 23\cdot x^6 + 17\cdot x^5 + 4\cdot x^4 + 17\cdot x^3 + 20\cdot x^2 \end{array} & 14\cdot x^3 + 19\cdot x^2 + 30\cdot x + 29 \\
\hline
\oplus\ \begin{array}{l} 30\cdot x^7 + 21\cdot x^6 + 7\cdot x^5 + 1\cdot x^4 + 21\cdot x^3 + 20\cdot x^2 + 0\cdot x \\ 30\cdot x^7 + 8\cdot x^6 + 4\cdot x^5 + 19\cdot x^4 + 26\cdot x^3 + 19\cdot x^2 + 29\cdot x \end{array} & \\
\hline
\oplus\ \begin{array}{l} 29\cdot x^6 + 3\cdot x^5 + 18\cdot x^4 + 15\cdot x^3 + 7\cdot x^2 + 29\cdot x + 0 \\ 29\cdot x^6 + 30\cdot x^5 + 15\cdot x^4 + 20\cdot x^3 + 18\cdot x^2 + 20\cdot x + 16 \end{array} & \\
\hline
29\cdot x^5 + 29\cdot x^4 + 27\cdot x^3 + 21\cdot x^2 + 9\cdot x + 16 &
\end{array}
$$

Рис. 3.37 Ілюстрація процесу ділення модифікованої вхідної кодової послідовності на твірний поліном для прикладу 3.38

$$
\begin{array}{l|l}
\oplus\ \begin{array}{l} 46\cdot x^9 + 53\cdot x^8 + 39\cdot x^7 + 27\cdot x^6 + 0\cdot x^5 + 0\cdot x^4 + 0\cdot x^3 + 0\cdot x^2 + 0\cdot x + 0 \\ 46\cdot x^9 + 43\cdot x^8 + 47\cdot x^7 + 11\cdot x^6 + 39\cdot x^5 + 4\cdot x^4 + 10\cdot x^3 \end{array} & x^6 + 61\cdot x^5 + 13\cdot x^4 + 55\cdot x^3 + 46\cdot x^2 + 48\cdot x + 59 \\
\hline
\oplus\ \begin{array}{l} 30\cdot x^8 + 8\cdot x^7 + 16\cdot x^6 + 39\cdot x^5 + 4\cdot x^4 + 10\cdot x^3 + 0\cdot x^2 \\ 30\cdot x^8 + 40\cdot x^7 + 16\cdot x^6 + 33\cdot x^5 + 35\cdot x^4 + 56\cdot x^3 + 47\cdot x^2 \end{array} & 46\cdot x^3 + 30\cdot x^2 + 32\cdot x + 2 \\
\hline
\oplus\ \begin{array}{l} 32\cdot x^7 + 0\cdot x^6 + 6\cdot x^5 + 39\cdot x^4 + 50\cdot x^3 + 47\cdot x^2 + 0\cdot x \\ 32\cdot x^7 + 2\cdot x^6 + 42\cdot x^5 + 13\cdot x^4 + 57\cdot x^3 + 40\cdot x^2 + 7\cdot x \end{array} & \\
\hline
\oplus\ \begin{array}{l} 2\cdot x^6 + 44\cdot x^5 + 42\cdot x^4 + 11\cdot x^3 + 7\cdot x^2 + 7\cdot x + 0 \\ 2\cdot x^6 + 57\cdot x^5 + 26\cdot x^4 + 45\cdot x^3 + 31\cdot x^2 + 35\cdot x + 53 \end{array} & \\
\hline
21\cdot x^5 + 48\cdot x^4 + 38\cdot x^3 + 24\cdot x^2 + 36\cdot x + 53 &
\end{array}
$$

Рис. 3.38 Ілюстрація процесу ділення модифікованої вхідної кодової послідовності на твірний поліном для прикладу 3.39

Приклад 3.40. Побудувати систематичний код Ріда – Соломона для кодової послідовності [12, 14, 7, 10] у полі Галуа $GF(2^4)$, вважаючи, що кількість помилок, які необхідно виправляти, становить $t = 2$.

Спочатку, з використанням співвідношення (3.200), знайдемо твірний поліном для такого коду:

$$g(x) = \prod_{i=1}^{4}\left(x - 2^i\right) = \left(x - 2^1\right)\cdot\left(x - 2^2\right)\cdot\left(x - 2^3\right)\cdot\left(x - 2^4\right) =$$

$$= (x-2)\cdot(x-4)\cdot(x-8)\cdot(x-3) = \left(x^2 + 6\cdot x + 8\right)\left(x^2 + 11\cdot x + 11\right) =$$

$$= x^4 + 13\cdot x^3 + 12\cdot x^2 + 8\cdot x + 7.$$

Цілком зрозуміло, що порядок цього поліному $g(x)$ дорівнює 4.

Тепер необхідно модифіковану вхідну послідовність $[12,14,7,10,0,0,0,0] =$ $= 12\cdot x^7 + 14\cdot x^6 + 7\cdot x^5 + 10\cdot x^4 + 0\cdot x^3 + 0\cdot x^2 + 0\cdot x + 0$ поділити стовпчиком, за схемою Горнера, на твірний поліном $g(x)$. Процес такого ділення показаний на рис. 3.39.

$$\begin{array}{l|l}
12\cdot x^7 + 14\cdot x^6 + 7\cdot x^5 + 10\cdot x^4 + 0\cdot x^3 + 0\cdot x^2 + 0\cdot x + 0 & x^4 + 13\cdot x^3 + 12\cdot x^2 + 8\cdot x + 7 \\
\oplus\ 12\cdot x^7 + 3\cdot x^6 + 15\cdot x^5 + 10\cdot x^4 + 2\cdot x^3 & \hline
\quad 13\cdot x^6 + 8\cdot x^5 + 0\cdot x^4 + 2\cdot x^3 + 0\cdot x^2 & 12\cdot x^3 + 13\cdot x^2 + 6\cdot x + 11 \\
\quad \oplus\ 13\cdot x^6 + 14\cdot x^5 + 3\cdot x^4 + 2\cdot x^3 + 5\cdot x^2 & \\
\hline
\quad\quad 6\cdot x^5 + 3\cdot x^4 + 0\cdot x^3 + 5\cdot x^2 + 0\cdot x & \\
\quad\quad \oplus\ 6\cdot x^5 + 8\cdot x^4 + 14\cdot x^3 + 5\cdot x^2 + 1\cdot x & \\
\hline
\quad\quad\quad 11\cdot x^4 + 14\cdot x^3 + 0\cdot x^2 + 1\cdot x + 0 & \\
\quad\quad\quad \oplus\ 11\cdot x^4 + 6\cdot x^3 + 13\cdot x^2 + 7\cdot x + 4 & \\
\hline
\quad\quad\quad\quad 8\cdot x^3 + 13\cdot x^2 + 6\cdot x + 4 &
\end{array}$$

Рис. 3.39 Ілюстрація процесу ділення модифікованої вхідної кодової послідовності на твірний поліном для прикладу 3.40

Видно, що остача від ділення на твірний поліном у цьому випадку становить $8\cdot x^3 + 13\cdot x^2 + 6\cdot x + 4$, тобто, кодова послідовність для коду Ріда – Соломона має наступний вигляд:

$$RS = 12\cdot x^7 + 14\cdot x^6 + 7\cdot x^5 + 10\cdot x^4 + 8\cdot x^3 + 13\cdot x^2 + 6\cdot x + 4 = [12, 14, 7, 10, 8, 13, 6, 4].$$

Приклад 3.41. Побудувати систематичний код Ріда – Соломона для

слова «Хакер», вважаючи, що всі літери цього слова подані через розширений код ASCII довжиною 1 байт, а кількість помилок, які виправляються, становить $t = 2$ [81].

Спосіб кодування інформації з використанням коду ASCII був розглянутий у підрозділі 1.3.2. Для повідомлення, яке розглядається, маємо наступні цифрові коди символів:

«Х» = 213; «а» = 224; «к» = 234; «е» = 229; «р» = 240.

Тобто, повідомлення «Хакер», яке необхідно закодувати, у числовій формі записується як $M(x)$ = [213, 224, 234, 229, 240].

Для формування коду Ріда – Соломона визначимо твірний поліном, з урахуванням того, що кількість помилок, які необхідно виправляти, складає $t = 2$. За визначених значень порядку поля $m = 8$ та кількості помилок $t = 2$, скориставшись співвідношенням (3.200), отримуємо наступний поліном $g(x)$:

$$g(x) = \prod_{i=1}^{4}\left(x - 2^i\right) = \left(x - 2^1\right)\cdot\left(x - 2^2\right)\cdot\left(x - 2^3\right)\cdot\left(x - 2^4\right) =$$

$$= (x-2)\cdot(x-4)\cdot(x-8)\cdot(x-16) = \left(x^2 + 6\cdot x + 8\right)\left(x^2 + 24\cdot x + 138\right) =$$

$$= x^4 + 30\cdot x^3 + 216\cdot x^2 + 231\cdot x + 116.$$

Поділимо модифіковану вхідну послідовність $M(x)$ = [213, 224, 234, 229, 240, 0, 0, 0, 0] на твірний поліном $x^4 + 30\cdot x^3 + 216\cdot x^2 + 231\cdot x + 116$ за схемою Горнера. Ітераційний процес цього ділення показаний на рис. 3.40.

Як видно з рис. 3.40, у цьому разі остача від ділення модифікованої вхідної кодової послідовності на твірний поліном становить $5\cdot x^3 + 61\cdot x^2 + 81\cdot x + 228$, тому для визначеного слова «Хакер» код Ріда – Соломона із виправленням подвійної помилки у полі Галуа $GF(2^8)$ записується наступним чином:

$$RS = 213\cdot x^8 + 224\cdot x^7 + 234\cdot x^6 + 229\cdot x^5 + 240\cdot x^4 + 5\cdot x^3 + 61\cdot x^2 + 81\cdot x + 228 =$$

$$= [213, 224, 234, 229, 240, 5, 61, 81, 228].$$

Оскільки алгебраїчні операції множення та ділення поліномів у полях Галуа реалізовані у комп'ютерних програмах, які наведені у додатку Н, тепер можна легко створити програму **RSC**, яка формує код Ріда – Соломона у полі Галуа $GF(2^m)$ за умови заданої вхідної послідовності, яку необхідно

закодувати, та визначеної максимальної кількості помилок, що виправляються. Комп'ютерний код такої програми, написаний мовою програмування системи MatLab, наведений у додатку О.

$$
\begin{array}{l|l}
213\cdot x^8+224\cdot x^7+234\cdot x^6+229\cdot x^5+240\cdot x^4+0\cdot x^3+0\cdot x^2+0\cdot x+0 & x^4+30\cdot x^3+216\cdot x^2+ \\
\oplus\ 213\cdot x^8+243\cdot x^7+146\cdot x^6+11\cdot x^5+126\cdot x^4 & \quad +231\cdot x+116 \\
\hline
19\cdot x^7+120\cdot x^6+238\cdot x^5+142\cdot x^4+0\cdot x^3 & 213\cdot x^4+19\cdot x^3+167\cdot x^2+ \\
\oplus\ 19\cdot x^7+223\cdot x^6+116\cdot x^5+226\cdot x^4+143\cdot x^3 & \quad +156\cdot x+154 \\
\hline
167\cdot x^6+154\cdot x^5+108\cdot x^4+143\cdot x^3+0\cdot x^2 & \\
\oplus\ 167\cdot x^6+6\cdot x^5+56\cdot x^4+192\cdot x^3+239\cdot x^2 & \\
\hline
156\cdot x^5+84\cdot x^4+79\cdot x^3+239\cdot x^2+0\cdot x & \\
\oplus\ 156\cdot x^5+206\cdot x^4+192\cdot x^3+248\cdot x^2+193\cdot x & \\
\hline
154\cdot x^4+143\cdot x^3+23\cdot x^2+193\cdot x+0 & \\
\oplus\ 154\cdot x^4+138\cdot x^3+42\cdot x^2+144\cdot x+228 & \\
\hline
5\cdot x^3+61\cdot x^2+81\cdot x+228 &
\end{array}
$$

Рис. 3.40 Ілюстрація процесу ділення модифікованої вхідної кодової послідовності на твірний поліном для прикладу 3.41

Як і попередні кодувальні програми, наведений у цьому посібнику, програма **RSC** одночасно призначена і для формування кодів Ріда – Соломона, і для декодування його кодових послідовностей. Для ініціалізації роботи цієї програми використовуються наступні параметри виклику:

vin – вхідний вектор, який являє собою або інформаційне слово, що необхідно закодувати, або закодовану послідовність символів, яку необхідно декодувати;

m – порядок поля Галуа, у якому формується код Ріда – Соломона;

ne – кількість помилок, які виявляються та виправляються;

nop – номер операції: 1 – кодування, 2 – декодування.

На вхідні параметри, які використовуються для запуску програми, накладаються відповідні обмеження. Як і для всіх алгебраїчних операцій у полі Галуа, порядок поля **m** може мати значення 3, 4, 5, 6, 7, 8, 12 або 16, а

параметр **ne**, який задає кількість помилок, що виправляються, може змінюватися в діапазоні від 1 до 6.

Частина програми, яка формує код Ріда – Соломона за заданим вхідним інформаційним словом, працює згідно з алгоритмом, наведеним на рис. 3.36. Для формування твірного поліному $g(x)$ використовується функція множення поліномів **prodpolgf2m**, а для алгебраїчних операції із вхідним словом у полі Галуа визначеного порядку – відповідно функції **divpolgf2m** та **sumgf2m**.

Алгоритм кодування, наведений на рис. 3.36, за умови наявності програмний модулів, призначених для виконання алгебраїчних операцій над поліномами у полях Галуа, є досить простим і його програмна реалізація не потребує подальшого окремого розгляду. Всі операції над векторами, які описують відповідні поліноми, реалізовані через векторні та матричні макрооперації системи науково-технічних розрахунків MatLab [13, 14].

На відміну від цього, алгоритми декодування кодів Ріда – Соломона за умови наявності помилок є вкрай складними і потребують більш поглибленого розгляду та досконалого аналізу [62, 63, 80, 81]. Зазвичай для декодування кодів Ріда – Соломона використовується ефективний алгоритм Берлекемпа – Мессі та теорема Форні, які були розглянуті у загальному вигляді у підрозділі 3.3.4.3. Особливості використання цього ітераційного алгоритму для декодування систематичних кодів Ріда – Соломона розглядатимуться у наступному підрозділі.

3.4.6 Декодування кодів Ріда – Соломона з використанням алгоритму Берлекемпа – Мессі та теореми Форні та відповідні приклади

Перед вивченням цього підрозділу необхідно повторити підрозділ 3.3.4.3

В цілому, згідно із теоретичними відомостями, наведеними у підрозділі 3.3.4.3 для кодів БЧХ, декодування кодових послідовностей групових кодів здійснюється за наступним алгоритмом [62, 63, 80, 81].

1. Обчислюється остача $R(x)$ від ділення вхідної кодової послідовності $F(x)$

на твірний поліном $g(x)$, який формується у полі Галуа визначеного порядку m за умови відомого значення кількості помилок, що виправляються, t.

2. Якщо остача від ділення у пункті 1 дорівнює 0, кодова послідовність вважається правильною. Тоді для отримання закодованого слова достатньо видалити із отриманої послідовності $2 \cdot t$ останніх розрядів.

3. Якщо остача від ділення у пункті 1 не дорівнює 0, кодова послідовність вважається спотвореною. У цьому випадку обчислюються значення поліному синдрому помилок $S(x)$. Для цього необхідно підставити до поліному, який відповідає отриманій кодовій послідовності $F(x)$, значення 2^i, де $i = 1, 2, \ldots, 2 \cdot t$.

4. За визначеними значеннями синдромів помилок S_1, S_2, ..., $S_{2 \cdot t}$, формується матриця синдромів помилок **M** та вектор синдромів **V**, які відповідають поліному локаторів помилок $\Lambda(x)$. Відповідні співвідношення мають наступний вигляд:

$$\mathbf{M} = \begin{pmatrix} S_{\tau} & S_{\tau-1} & S_{\tau-2} & \ldots & S_1 \\ S_{\tau+1} & S_{\tau} & S_{\tau-1} & \ldots & S_2 \\ S_{\tau+2} & S_{\tau+1} & S_{\tau} & \ldots & S_3 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ S_{2\cdot\tau-1} & S_{2\cdot\tau-2} & S_{2\cdot\tau-3} & \ldots & S_{2\cdot\tau-\tau} \end{pmatrix}; \qquad \mathbf{V} = \begin{pmatrix} S_{\tau+1} \\ S_{\tau+2} \\ S_{\tau+3} \\ \ldots \\ S_{2\cdot\tau} \end{pmatrix}. \qquad (3.220)$$

5. Розв’язується матричне рівняння

$$\Lambda(x) = \mathbf{M}^{-1} \cdot \mathbf{V}, \qquad (3.221)$$

де $\Lambda(x)$ – поліном локаторів помилок, який розглядався у підрозділі 3.3.4.3 для кодів БЧХ та визначається співвідношенням (3.80). У конкретному випадку, для кодів Ріда – Соломона, можна переписати поліном локаторів помилок $\Lambda(x)$ для поля Галуа $GF(2^m)$ у вигляді:

$$\Lambda(x) = 1 + \sum_{i=1}^{\tau} \Lambda_i \cdot x^i = \prod_{i=1}^{\nu} \left(1 + x^i \cdot 2^{u_i}\right). \qquad (3.222)$$

6. Пошук коренів полінома локаторів помилок $\Lambda(x)$ методом перебирання всіх елементів поля Галуа, які відповідаютсья номерам розрядів кодвої комбінації. Слід зазначити, що кожен елемент підставляється до поліному лише один раз і вважається коренем поліному локаторів помилок, якщо значення поліному для цього елементу дорівнює нулю. Якщо кількість

знайдених коренів дорівнює кратності помилок t, рівняння $\Lambda(x)=0$ вважається розв'язаним та пошук коренів припиняється.

7. Обчислення похідної від полінома локаторів помилок $\Lambda'(x)$. У загальному вигляді вираз для похідної функції (3.222) був сформований у підрозділі 3.3.4.3 під час доведення теореми Форні 3.7, проте для кодів Ріда – Соломона він має свої особливості.

8. Пошук значень елементів вектора помилок з використанням теореми Форні 3.7 та співвідношення (3.84).

9. Формування вектора помилок $E(x)$ на основі коренів полінома локатора помилок та знайдених значень елементів вектора помилок.

10. Виправлення спотвореної кодової комбінації $C(x)$ через сумування її за модулем два із вектором помилок $E(x)$, в результаті чого формується початковий, неспотворений код $F(x)$, тобто:

$$F(x)=C(x)\oplus E(x). \qquad (3.223)$$

Блок-схема узагальненого алгоритму декодування кодів Ріда – Соломона показана на рис. 3.41.

Хоча з теоретичної точки зору алгоритм Берлекемпа – Мессі та теорема Форні були повністю описані у підрозділі 3.3.4.3, реалізація цих алгоритмів для декодування кодів Ріда – Соломона має свої особливості, які треба проаналізувати більш досконало.

Із співвідношення (3.222), отриманого для поліному локаторів помилок, можна зробити дуже цікавий висновок про те, що коренями поліному $\Lambda(x)$ є значення елементів поля Галуа 2^{-u_1}, 2^{-u_2}, ..., $2^{-\tau}$. Тут слід зазначити, що хоча, згідно із теорією груп, від'ємних чисел у полях Галуа $GF(2^m)$ не існує, для поліномів над елементами полів Галуа дуже зручно використовувати поняття від'ємної степені. Загалом це поняття пов'язане із алгебраїчною операцію ділення та із способом обчислення оберненого елемента відповідно до співвідношення (3.213) [48]. Надамо відповідне визначення [62, 63, 80, 81].

Визначення 3.12. Алгебраїчна операція піднесення до від'ємної степені $-u$ числа 2 у полі Галуа $GF(2^m)$ еквівалентна піднесенню числа 2 до степені $(2^m - 1 - u)$, тобто:

$$2^{-u} = \frac{1}{2^u} = 2^{(2^m-1)-u}. \qquad (3.224)$$

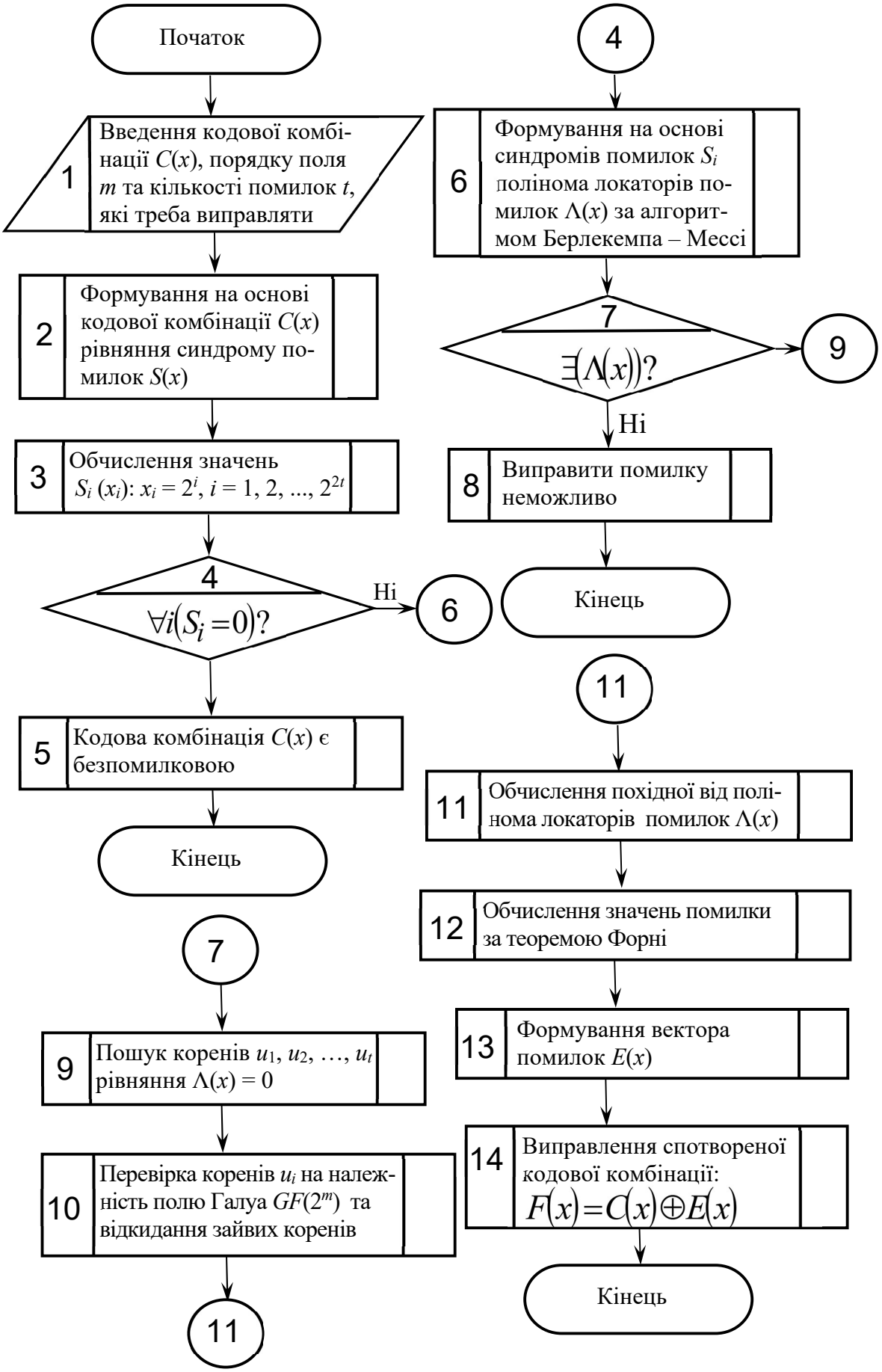


Рис. 3.41 Блок-схема алгоритму декодування кодів Ріда – Соломона

Оскільки обчислення степеневої функції від від'ємного аргументу є важливою алгебраїчною операцією, яка ефективно використовується для декодування кодів Ріда – Соломона, ця операція також реалізована у програмному модулі **powgf2m**, який наведений у додатку Н. Там же наведені приклади обчислення значень степеневої функції за умови її від'ємного аргументу.

Слід відзначити, що зазвичай перевіряти всі елементи поля Галуа $GF(2^m)$ на корені поліному локаторів помилок $\Lambda(x)$ немає необхідності, оскільки можливі значення номерів помилкових розрядів у коді Ріда – Соломона не перевищують розрядності коду n. Розряди кодової комбінації нумеруються як в поліномах, тобто, першому розряду відповідає номер 0, а останньому – $n - 1$. Тому до поліному локаторів помилок $\Lambda(x)$ достатньо підставити n значень: 2^0, 2^{-1}, ..., $2^{-(n-1)}$. Із цього принципу пошуку помилкових розрядів випливає важливе обмеження на довжину коду Ріда – Соломона: максимальна довжина коду не повинна перевищувати значення $2^m - 1$, де m – порядок поля Галуа.

Матричну систему рівнянь (3.221), з урахуванням співвідношень (3.220), можна переписати наступним чином [62, 63, 80, 81]:

$$S_j = \sum_{i=1}^{\tau} \Lambda_i \cdot S_j,$$
$$j = \tau + 1 \ldots 2 \cdot \tau.$$

Слід відзначити, що для розв'язування матричної системи рівнянь (3.221) може бути використаний будь-який із методів матричного аналізу, розглянутих у другій частині посібника. Тут головною задачею є пошук зворотної матриці $\mathbf{M}^{-1}$, яку можна знайти через обчислення визначників, через множення рядків на число, їх сумування та формування діагональної матриці, або через обчислення власних чисел матриці [48]. Єдиною вимогою є те, що всі алгебраїчні операції над елементами матриці **M** необхідно проводити у полі Галуа $GF(2^m)$ [62, 63, 80, 81]. Проте у теорії кодування найбільш ефективним методом пошуку розв'язку рівняння (3.221) вважається алгоритм Берлекемпа – Мессі. Теоретично обґрунтовано, що у разі

використання цього алгоритму кількість ітерацій для формування поліному локаторів помилок не перевищує значення $2 \cdot t$ [62, 63, 80, 81]. Тому, незважаючи на те, що коди Ріда – Соломона використовувались в обчислювальній техніці та у системах зв'язку ще з 1960 року, коли вони були вперше запропоновані, спочатку їхнє використання було не дуже ефективним, оскільки формувалися лише синдроми наявності помилок, а знайти їх та виправити не вдавалося. І лише починаючи з 1968 – 1969 років, коли Берлекемп та Мессі запропонували свій простий та зручний алгоритм декодування кодових послідовностей групових кодів, який дозволив знаходити та виправляти помилки у них, такі коди стали дійсно широко та ефективно використовуватися у цифровій кодувальній електронній апаратурі, зокрема в обчислювальній техніці та у системах зв'язку. Берлекемп та Мессі довели наступну теорему [52, 62, 63].

Теорема 3.11. Якщо існує однозначний розв'язок лінійної системи рівнянь (3.220) в області значень τ $1 \le \tau \le t$, степінь поліному локаторів помилок $\Lambda(x)$ для цього розв'язку є мінімальною.

Згідно із цією теоремою пошук розв'язку системи рівнянь (3.220), (3.221), зводиться до розв'язування наступної задачі оптимізації:

$$\left(\begin{array}{c} S_j = \sum\limits_{i=1}^{\tau} \Lambda_i \cdot S_j, \\ \tau = \min\left(\deg\left(\Lambda(x)\right)\right), \\ j = \tau + 1 \ldots 2 \cdot \tau. \end{array} \right) \rightarrow \left(\begin{array}{c} \tau \\ \Lambda(x) \end{array} \right). \qquad (3.225)$$

Алгоритм Берлекемпа – Мессі, узагальнена блок-схема якого наведена на рис. 3.8, є вкрай непростим. Сутність його полягає у тому, що, починаючи з тривіального незвідного поліному нульового порядку $\Lambda(x) = 1$, обчислюється відхилення цього поліному відносно першого рівняння системи (3.220). Далі, з урахуванням цього відхилу, проводиться коректування поліному локаторів помилок, а у разі необхідності нарощується його степінь. На наступній ітерації все повторюється таким чином, щоб, за умови відповідності створеного поліному попереднім рівнянням, він відповідав також поточному рівнянню

системи (3.220). Ітераційний процес проводиться то тих пір, доки не буде розглянуто останнє рівняння системи (3.220). І оскільки алгоритм формування поліному побудований таким чином, що степінь його нарощується лише у разі необхідності, такий поліном завжди буде задовольняти рівнянню мінімізації (3.225).

Можливими є два результати обчислення коефіцієнтів локаторів помилок з використанням алгоритму Берлекемпа – Мессі. Якщо кількість спотворених бітів є більшою за максимальну кількість помилок, які виправляються, тоді степінь поліному локаторів не співпадає із передбаченою кількістю спотворених бітів. У цьому випадку вважається, що виправити помилки у коді неможливо.

Більш цікавим з практичної точки зору є інший випадок, коли знайдений поліном локаторів помилок $\Lambda(x)$ відповідає кратності помилок τ. Тоді через перебирання значень 2^0, 2^{-1}, ..., $2^{-(n-1)}$ та підстановки їх до поліному $\Lambda(x)$ шукаються номера помилкових розрядів кодової комбінації.

Головними внутрішніми параметрами алгоритму Берлекемпа – Мессі, які використовуються для пошуку коефіцієнтів поліному локаторів помилок, є наступні:

r – номер ітерації алгоритму: початкове значення цього параметру $r = 0$;

Δ_r – значення відхилення, яке розраховується як сума лівої та правої частин відповідного рівняння та дорівнює нулю, якщо поліном відповідає цьому рівнянню;

m – номер ітерації r, на якому останній раз змінювався поліном, що створюється;

L – число членів у лівій частині рівняння (3.224), або число стовпчиків у матриці **M**, яке завжди відповідає кількості спотворених байтів τ;

$B(x)$ – поліном модифікації, який добавляється до полінома локаторів помилок, щоб він задовольняв як попереднім рівнянням системи, так і поточному та забезпечував мінімальне відхилення.

Сутність алгоритму Берлекемпа – Мессі полягає у тому, що до сформованого

поліному $T(x)$ на поточній ітерації r додається новий член $\Delta_r \cdot B(x)$, в результаті чого сформований поліном має мінімальне відхилення для всіх розглянутих рівнянь системи (3.224), тобто на кожному ітераційному кроці r виконується умова мінімізації (3.225). На останній ітерації, коли розглянуті всі рівняння, $T(x) = \Lambda(x)$.

У разі, якщо помилкові розряди успішно знайдені, необхідно знайти поліном величин помилок з використанням теореми Форні через розв'язування поліноміального рівняння (3.81), тобто:

$$\Omega(x) = \mathrm{mod}\left((S(x) \cdot \Lambda(x)), \left(x^{2t}\right)\right). \qquad (3.226)$$

Тут слід відзначити, що в рівнянні (3.226) операція обчислення остачі від добутку поліномів $\left(S(x) \cdot \Lambda(x)\right)$ за модулем $\left(x^{2t}\right)$ просто відповідає відкиданню всіх складових поліному $\Omega(x)$, степінь яких перевищує значення $2t$ або дорівнює цьому значенню. Тобто, для поліному $\Omega(x)$ завжди виконується тотожність:

$$\deg(\Omega(x)) = 2 \cdot t - 1. \qquad (3.227)$$

Тому у разі програмної реалізації алгоритму пошуку поліному величин помилок $\Omega(x)$ достатньо знайти добуток поліномів $\left(S(x) \cdot \Lambda(x)\right)$ за правилами алгебри полів Галуа та видалити із отриманого результату старші розряди, починаючи із степені $2t$. З урахуванням того, що у системі MatLab коефіцієнти поліномів записуються як вектори, необхідно видалити $k = n - 2t - 1$ молодших розрядів отриманого вектора, пам'ятаючи про те, що у числах ми використовуємо порядок нумерування розрядів зправа-наліво, а у векторах – зліва-направо [13, 14]. Тобто, ділити результат добутку поліномів $\left(S(x) \cdot \Lambda(x)\right)$ на поліном $\left(x^{2t}\right)$ за правилами алгебри полів Галуа та брати остачу від цього ділення немає необхідності, це лише ускладнює алгоритм пошуку поліному величин помилок $\Omega(x)$ [81].

Похідна від поліному локаторів помилок $\Lambda'(x)$ обчислюється за звичайними правилами диференціювання поліноміальних функцій, єдина

відмінність полягає у тому, що всі алгебраїчні операції виконуються у визначеному полі Галуа. У загальному випадку для обчислення поліному $\Lambda'(x)$ можна записати наступний алгебраїчний вираз [81]:

$$\Lambda'(x) = \Lambda_1 \oplus \Lambda_3 \cdot x^2 \oplus \ldots \oplus \begin{cases} \Lambda_\tau \cdot x^{\tau-1}, & \tau \bmod 2 = 1; \\ \Lambda_{\tau-1} \cdot x^{\tau-2}, & \tau \bmod 2 = 0. \end{cases} \quad (3.228)$$

Тоді остаточні значення помилок обчислюються, згідно із теоремою Форні та співвідношенням (3.84), наступним чином:

$$\nu_l = \frac{\Omega\left(2^{-u_l}\right)}{\Lambda'\left(2^{-u_l}\right)}, l = 1 \ldots \tau . \quad (3.229)$$

Оскільки тепер нам відомі як номери помилкових розрядів u_l, так із значення елементів вектору помилок ν_l, які цим розрядам відповідають, можна сформувати вектор помилки $E(x)$ та записати його у вигляді полінома у полі Галуа $GF(2^m)$ наступним чином:

$$E(x) = \sum_{i=1}^{\tau} \nu_l \cdot x^{u_l} . \quad (3.230)$$

За умови відомого вектора помилок $E(x)$ остаточну виправлену кодову комбінацію $F(x)$ можна знайти з використанням співвідношення (3.223). Тобто, задачу декодування коду Ріда – Соломона за умови наявності у ньому помилок кратності τ можна вважати розв'язаною.

Щодо похідних від поліному локаторів помилок $\Lambda'(x)$, заданих співвідношенням (3.228), існує просте правило для їхнього обчислення. Для парних степенів похідна завжди дорівнює 0, а для непарних – степені, зменшеній на – 1. Наприклад, $\Lambda'\left(x^4\right) = 0$, $\Lambda'\left(x^5\right) = x^4$. Тобто, у полях Галуа, на відміну від звичайної арифметики, похідні поліноміальних складових не множаться на коефіцієнт $n - 1$, де n – показник степені [81, 82]. Наприклад, для поліномів $\Lambda(x)$ від першого до шостого порядку можна записати наступні логічні вирази:

$$\begin{aligned}
&\left(\deg\left(\Lambda(x)\right)=1\right)\vee\left(\deg\left(\Lambda(x)\right)=2\right)\Rightarrow\Lambda'(x)=\lambda_1;\\
&\left(\deg\left(\Lambda(x)\right)=3\right)\vee\left(\deg\left(\Lambda(x)\right)=4\right)\Rightarrow\Lambda'(x)=\lambda_1\oplus\lambda_3\cdot x^2;\\
&\left(\left(\deg\left(\Lambda(x)\right)=5\right)\vee\left(\deg\left(\Lambda(x)\right)=6\right)\right)\Rightarrow\Lambda'(x)=\lambda_1\oplus\lambda_3\cdot x^2\oplus\lambda_5\cdot x^4.
\end{aligned}\qquad(3.231)$$

Описана закономірність формування похідних від поліномів локаторів помилок високих порядків цілком зрозуміла із наведених співвідношень (3.231), проте слід відзначити, що коди Ріда – Соломона із можливістю виявлення більше шести помилок є вкрай надлишковими, і з цієї причини на практиці вони майже не використовуються.

Загалом наведені у цьому підрозділі співвідношення (3.220) – (3.231) послідовно використовуються для реалізації процесу декодування кодів Ріда – Соломона із виявленням та виправленням помилок у ньому, згідно із алгоритмом, який був наведений на рис. 3.41. У разі наявності у коді одиночної помилки цей алгоритм значно спрощується, відповідні аналітичні співвідношення будуть наведені у цьому підрозділі далі.

Розглянемо тепер обмеження, які накладаються на параметри блокових кодів Ріда – Соломона. Ці обмеження головним чином стосуються зв’язку між довжиною пакетів B та кількістю помилок t, які можна виявити та виправити у заданому полі Галуа $GF(2^m)$. Базовим тут є співвідношення (3.204), яке свідчить про те, що кількість помилок, які виявляються та виправляються, має бути меншим за довжину пакетів B. Іншими обмеженнями, обумовленими особливостями описаного алгоритму декодування, є наступні.

1. Кількість помилок у коді, які виявляються та виправляються, помножена на 2, тобто значення $2\cdot t$, яке відповідає кількості контрольних символів r, не має перевищувати значення максимального елемента поля $e_{\max}=2^m-1$, тобто:

$$r=2\cdot t\le 2^m-1.\qquad(3.232)$$

2. Загальна довжина кодової комбінації B не має перевищувати значення максимального елементу поля $e_{\max}$, тобто:

$$B>2\cdot t\le 2^m-1.\qquad(3.233)$$

Зрозуміло, що, з урахуванням базового співвідношення між параметрами коду (3.204), нерівність (3.233) є більш строгою, із неї безпосередньо випливає співвідношення (3.232). Що до обмеження, пов'язаного із нерівністю (3.232), воно безпосередньо випливає із алгоритму формування кодів Ріда – Соломона через множення поліноміальних функцій у полі Галуа $GF(2^m)$, який був описаний у підрозділі 3.4.5 та наведений на рис. 3.36. Дійсно, формування твірного поліному $g(x)$ через множення одночленів $f(x) = \left(x \oplus 2^i\right)$, згідно із співвідношенням (3.200), є можливим лише за умови $i \le 2^m - 1$, оскільки показник степені у функції $f(x)$ має бути меншим за значення максимального елементу поля Галуа $e_{\max}$. Враховуючи, що, відповідно до співвідношення (3.200), $i_{\max} = 2 \cdot t$, отримуємо нерівність (3.232).

Щодо нерівності (3.233), вона випливає із того, що, згідно із описаним вище алгоритмом декодування, всі розряди кодової послідовності мають бути пронумеровані у полі Галуа $GF(2^m)$. У противному випадку неможливо визначити номери помилкових розрядів u_l. Враховуючи те, що обмеження (3.233) є більш строгим, ніж (3.232), ці співвідношення можна об'єднати та записати єдине, узагальнене обмеження на довжину блочного коду Ріда – Соломона наступним чином:

$$B = k + 2 \cdot t \le 2^m - 1, \qquad (3.234)$$

де k – довжина інформаційного слова, яке кодується.

Якщо співвідношення (3.234) не виконується, необхідно або зменшувати довжину блоків кодових комбінацій B через зменшення довжини інформаційних слів, або зменшувати кількість помилок t, які виявляються та виправляються.

Сформулюємо розглянуті обмеження як властивості блокових кодів Ріда – Соломона.

Властивість 3.1. Довжина пакетів B у блоковому коді Ріда – Соломона не повинна перевищувати значення максимального елемента поля Галуа $GF(2^m)$ $e_{\max} = 2^m - 1$.

Властивість 3.2. (Наслідок властивості 3.1). Кількість контрольних розрядів r у блоковому коді Ріда – Соломона, яка визначається як подвійне значення від кількості помилок t, які виявляються та виправляються, тобто, $r = 2t$, не повинна перевищувати значення максимального елемента поля Галуа $GF(2^m)$ $e_{\max} = 2^m - 1$.

Зрозуміло, що властивість 3.1 обов'язково необхідно враховувати у разі формування блокових кодів Ріда – Соломона, як ручним способом, так і з використанням програмних засобів. Щодо властивості 3.2, вона завжди виконується у разі, якщо виконана властивість 3.1.

Розглянемо декілька прикладів декодування кодів Ріда – Соломона у разі наявності одиночних та кратних помилок.

Приклад 3.42. У прикладі 3.41 був побудований код Ріда – Соломона для слова «Хакер». Припустимо, що під час передавання інформації виникли помилки у дев'ятому та сьомому байтах кодової послідовності, і, випадково, замість слова «Хакер» було передане інше слово, «Ламер», проте всі інші байти прийнятої кодової послідовності є вірними. Сформувати поліном синдрому помилок та обчислити його значення для правильної та спотвореної кодової комбінації. З використанням алгоритму Берлекемпа – Мессі та теореми Форні розшифрувати прийняту кодову послідовність та знайти в ній помилки [81].

Будемо розв'язувати задачу послідовно.

1. Сформуємо спотворену кодову послідовність.

У прикладі 3.41 для слова «Хакер» був отриманий код RS = [213, 224, 234, 229, 240, 5, 61, 81, 228], сформований у форматі ASCII. Зважаючи на те, що ASCII-код літери «Л» становить 203, а ASCII-код літери «м» – 236, спотворена кодова послідовність має наступний вигляд $\mathbf{C}$ = [203, 224, 236, 229, 240, 5, 61, 81, 228].

2. Обчислимо значення поліному синдрому помилок $S(x)$ у полі Галуа $GF(2^8)$.

Для прикладу, який розглядається, маємо.

$S(x) = 203 \cdot x^8 + 224 \cdot x^7 + 236 \cdot x^6 + 229 \cdot x^5 + 240 \cdot x^4 + 5 \cdot x^3 + 61 \cdot x^2 + 81 \cdot x + 228.$

Оскільки сформований код Ріда – Соломона призначений для виявлення та виправлення подвійних помилок, необхідно обчислити значення суми $S(x)$ для таких аргументів: $x_1 = 2^1 = 2$, $x_2 = 2^2 = 4$, $x_3 = 2^3 = 8$ та $x_4 = 2^4 = 16$. Відповідно маємо:

$$S_1(x_1) = 203 \cdot 2^8 \oplus 224 \cdot 2^7 \oplus 236 \cdot 2^6 \oplus 229 \cdot 2^5 \oplus 240 \cdot 2^4 \oplus 5 \cdot 2^3 \oplus 61 \cdot 2^2 \oplus$$
$$\oplus 81 \cdot 2 \oplus 228 = 250 \oplus 89 \oplus 133 \oplus 241 \oplus 187 \oplus 40 \oplus 244 \oplus$$
$$\oplus 162 \oplus 228 = 163 \oplus 133 \oplus 241 \oplus 187 \oplus 40 \oplus 244 \oplus 162 \oplus$$
$$\oplus 228 = 38 \oplus 241 \oplus 187 \oplus 40 \oplus 244 \oplus 162 \oplus 228 =$$
$$= 215 \oplus 187 \oplus 40 \oplus 244 \oplus 162 \oplus 228 = 108 \oplus 40 \oplus 244 \oplus$$
$$\oplus 162 \oplus 228 = 68 \oplus 244 \oplus 162 \oplus 228 = 176 \oplus 162 \oplus 228 =$$
$$= 18 \oplus 228 = 246.$$

$$S_2(x_2) = 203 \cdot \left(2^2\right)^8 \oplus 224 \cdot \left(2^2\right)^7 \oplus 236 \cdot \left(2^2\right)^6 \oplus 229 \cdot \left(2^2\right)^5 \oplus 240 \cdot \left(2^2\right)^4 \oplus$$
$$\oplus 5 \cdot \left(2^2\right)^3 \oplus 61 \cdot \left(2^2\right)^2 \oplus 81 \cdot \left(2^2\right) \oplus 228 = 203 \cdot 2^{16} \oplus 224 \cdot 2^{14} \oplus$$
$$\oplus 236 \cdot 2^{12} \oplus 229 \cdot 2^{10} \oplus 240 \cdot 2^8 \oplus 5 \cdot 2^6 \oplus 61 \cdot 2^4 \oplus 81 \cdot 2^2 \oplus 228 =$$
$$= 203 \cdot 76 \oplus 224 \cdot 19 \oplus 236 \cdot 205 \oplus 229 \cdot 116 \oplus 240 \cdot 29 \oplus 5 \cdot 64 \oplus$$
$$\oplus 61 \cdot 16 \oplus 81 \cdot 4 \oplus 228 = 173 \oplus 155 \oplus 218 \oplus 75 \oplus 127 \oplus 93 \oplus 247 \oplus$$
$$\oplus 89 \oplus 228 = 54 \oplus 218 \oplus 75 \oplus 127 \oplus 93 \oplus 247 \oplus 89 \oplus 228 = 236 \oplus$$
$$\oplus 75 \oplus 127 \oplus 93 \oplus 247 \oplus 89 \oplus 228 = 167 \oplus 127 \oplus 93 \oplus 247 \oplus 89 \oplus$$
$$\oplus 228 = 216 \oplus 93 \oplus 247 \oplus 89 \oplus 228 = 133 \oplus 247 \oplus 89 \oplus 228 =$$
$$= 114 \oplus 89 \oplus 228 = 43 \oplus 228 = 207 .$$

$$S_3(x_3) = 203 \cdot \left(2^3\right)^8 \oplus 224 \cdot \left(2^3\right)^7 \oplus 236 \cdot \left(2^3\right)^6 \oplus 229 \cdot \left(2^3\right)^5 \oplus 240 \cdot \left(2^3\right)^4 \oplus$$
$$\oplus 5 \cdot \left(2^3\right)^3 \oplus 61 \cdot \left(2^3\right)^2 \oplus 81 \cdot \left(2^3\right) \oplus 228 = 203 \cdot 2^{24} \oplus 224 \cdot 2^{21} \oplus$$
$$\oplus 236 \cdot 2^{18} \oplus 229 \cdot 2^{15} \oplus 240 \cdot 2^{12} \oplus 5 \cdot 2^9 \oplus 61 \cdot 2^6 \oplus 81 \cdot 2^3 \oplus 228 =$$
$$= 203 \cdot 143 \oplus 224 \cdot 117 \oplus 236 \cdot 45 \oplus 229 \cdot 38 \oplus 240 \cdot 205 \oplus 5 \cdot 58 \oplus$$
$$\oplus 61 \cdot 64 \oplus 81 \cdot 8 \oplus 228 = 32 \oplus 18 \oplus 132 \oplus 149 \oplus 163 \oplus 210 \oplus 251 \oplus$$
$$\oplus 178 \oplus 228 = 50 \oplus 132 \oplus 149 \oplus 163 \oplus 210 \oplus 251 \oplus 178 \oplus 228 =$$
$$= 182 \oplus 149 \oplus 163 \oplus 210 \oplus 251 \oplus 178 \oplus 228 = 35 \oplus 163 \oplus 210 \oplus$$
$$\oplus 251 \oplus 178 \oplus 228 = 128 \oplus 210 \oplus 251 \oplus 178 \oplus 228 = 82 \oplus 251 \oplus$$

$$\oplus 178 \oplus 228 = 169 \oplus 178 \oplus 228 = 27 \oplus 228 = 255 .$$

$$S_4(x_4) = 203 \cdot \left(2^4\right)^8 \oplus 224 \cdot \left(2^4\right)^7 \oplus 236 \cdot \left(2^4\right)^6 \oplus 229 \cdot \left(2^4\right)^5 \oplus 240 \cdot \left(2^4\right)^4 \oplus$$

$$\oplus 5 \cdot \left(2^4\right)^3 \oplus 61 \cdot \left(2^4\right)^2 \oplus 81 \cdot \left(2^4\right) \oplus 228 = 203 \cdot 2^{32} \oplus 224 \cdot 2^{28} \oplus$$

$$\oplus 236 \cdot 2^{24} \oplus 229 \cdot 2^{20} \oplus 240 \cdot 2^{16} \oplus 5 \cdot 2^{12} \oplus 61 \cdot 2^8 \oplus 81 \cdot 2^4 \oplus 228 =$$

$$= 203 \cdot 157 \oplus 224 \cdot 24 \oplus 236 \cdot 143 \oplus 229 \cdot 180 \oplus 240 \cdot 76 \oplus 5 \cdot 205 \oplus$$

$$\oplus 61 \cdot 29 \oplus 81 \cdot 16 \oplus 228 = 135 \oplus 245 \oplus 154 \oplus 87 \oplus 226 \oplus 222 \oplus$$

$$\oplus 203 \oplus 121 \oplus 228 = 114 \oplus 154 \oplus 87 \oplus 226 \oplus 222 \oplus 203 \oplus 121 \oplus$$

$$\oplus 228 = 232 \oplus 87 \oplus 226 \oplus 222 \oplus 203 \oplus 121 \oplus 228 = 191 \oplus 226 \oplus$$

$$\oplus 222 \oplus 203 \oplus 121 \oplus 228 = 93 \oplus 222 \oplus 203 \oplus 121 \oplus 228 =$$

$$= 131 \oplus 203 \oplus 121 \oplus 228 = 72 \oplus 121 \oplus 228 = 49 \oplus 228 = 213 .$$

Тобто, у полі Галуа $GF(2^8)$ маємо такі значення поліному синдрому помилки: $S_1(x_1) = 246 \neq 0;\ S_2(x_2) = 207 \neq 0;\ S_3(x_3) = 255 \neq 0;\ S_4(x_4) = 213 \neq 0$, що свідчить про наявність помилок у кодовій комбінації.

Цікавою з теоретичної точки зору є перевірка значень поліному синдрому помилки для правильної кодової комбінації, тому проведемо відповідні розрахунки і для неї. Для кодової послідовності *RS* = [213, 224, 234, 229, 240, 5, 61, 81, 228], отриманої у прикладі 3.41, відповідно, маємо такі результати.

$$S_1(x_1) = 213 \cdot 2^8 \oplus 224 \cdot 2^7 \oplus 234 \cdot 2^6 \oplus 229 \cdot 2^5 \oplus 240 \cdot 2^4 \oplus 5 \cdot 2^3 \oplus 61 \cdot 2^2 \oplus$$

$$\oplus 81 \cdot 2 \oplus 228 = 145 \oplus 89 \oplus 24 \oplus 241 \oplus 187 \oplus 40 \oplus 244 \oplus$$

$$\oplus 162 \oplus 228 = 200 \oplus 24 \oplus 241 \oplus 187 \oplus 40 \oplus 244 \oplus 162 \oplus$$

$$\oplus 228 = 208 \oplus 241 \oplus 187 \oplus 40 \oplus 244 \oplus 162 \oplus 228 =$$

$$= 33 \oplus 187 \oplus 40 \oplus 244 \oplus 162 \oplus 228 = 154 \oplus 40 \oplus 244 \oplus$$

$$\oplus 162 \oplus 228 = 178 \oplus 244 \oplus 162 \oplus 228 = 70 \oplus 162 \oplus 228 =$$

$$= 228 \oplus 228 = 0.$$

$$S_2(x_2) = 213 \cdot \left(2^2\right)^8 \oplus 224 \cdot \left(2^2\right)^7 \oplus 234 \cdot \left(2^2\right)^6 \oplus 229 \cdot \left(2^2\right)^5 \oplus 240 \cdot \left(2^2\right)^4 \oplus$$

$$\oplus 5 \cdot \left(2^2\right)^3 \oplus 61 \cdot \left(2^2\right)^2 \oplus 81 \cdot \left(2^2\right) \oplus 228 = 213 \cdot 2^{16} \oplus 224 \cdot 2^{14} \oplus$$

$$\oplus 234 \cdot 2^{12} \oplus 229 \cdot 2^{10} \oplus 240 \cdot 2^8 \oplus 5 \cdot 2^6 \oplus 61 \cdot 2^4 \oplus 81 \cdot 2^2 \oplus 228 =$$

$$= 213 \cdot 76 \oplus 224 \cdot 19 \oplus 234 \cdot 205 \oplus 229 \cdot 116 \oplus 240 \cdot 29 \oplus 5 \cdot 64 \oplus$$
$$\oplus 61 \cdot 16 \oplus 81 \cdot 4 \oplus 228 = 246 \oplus 155 \oplus 78 \oplus 75 \oplus 127 \oplus 93 \oplus 247 \oplus$$
$$\oplus 89 \oplus 228 = 109 \oplus 78 \oplus 75 \oplus 127 \oplus 93 \oplus 247 \oplus 89 \oplus 228 =$$
$$= 35 \oplus 75 \oplus 127 \oplus 93 \oplus 247 \oplus 89 \oplus 228 = 104 \oplus 127 \oplus 93 \oplus 247 \oplus$$
$$\oplus 89 \oplus 228 = 23 \oplus 93 \oplus 247 \oplus 89 \oplus 228 = 74 \oplus 247 \oplus 89 \oplus 228 =$$
$$= 189 \oplus 89 \oplus 228 = 228 \oplus 228 = 0.$$

$$S_3(x_3) = 213 \cdot \left(2^3\right)^8 \oplus 224 \cdot \left(2^3\right)^7 \oplus 234 \cdot \left(2^3\right)^6 \oplus 229 \cdot \left(2^3\right)^5 \oplus 240 \cdot \left(2^3\right)^4 \oplus$$
$$\oplus 5 \cdot \left(2^3\right)^3 \oplus 61 \cdot \left(2^3\right)^2 \oplus 81 \cdot \left(2^3\right) \oplus 228 = 213 \cdot 2^{24} \oplus 224 \cdot 2^{21} \oplus$$
$$\oplus 234 \cdot 2^{18} \oplus 229 \cdot 2^{15} \oplus 240 \cdot 2^{12} \oplus 5 \cdot 2^9 \oplus 61 \cdot 2^6 \oplus 81 \cdot 2^3 \oplus 228 =$$
$$= 213 \cdot 143 \oplus 224 \cdot 117 \oplus 234 \cdot 45 \oplus 229 \cdot 38 \oplus 240 \cdot 205 \oplus 5 \cdot 58 \oplus$$
$$\oplus 61 \cdot 64 \oplus 81 \cdot 8 \oplus 228 = 49 \oplus 18 \oplus 106 \oplus 149 \oplus 163 \oplus 210 \oplus 251 \oplus$$
$$\oplus 178 \oplus 228 = 35 \oplus 106 \oplus 149 \oplus 163 \oplus 210 \oplus 251 \oplus 178 \oplus 228 =$$
$$= 73 \oplus 149 \oplus 163 \oplus 210 \oplus 251 \oplus 178 \oplus 228 = 220 \oplus 163 \oplus 210 \oplus$$
$$\oplus 251 \oplus 178 \oplus 228 = 127 \oplus 210 \oplus 251 \oplus 178 \oplus 228 = 173 \oplus 251 \oplus$$
$$\oplus 178 \oplus 228 = 86 \oplus 178 \oplus 228 = 228 \oplus 228 = 0.$$

$$S_4(x_4) = 213 \cdot \left(2^4\right)^8 \oplus 224 \cdot \left(2^4\right)^7 \oplus 234 \cdot \left(2^4\right)^6 \oplus 229 \cdot \left(2^4\right)^5 \oplus 240 \cdot \left(2^4\right)^4 \oplus$$
$$\oplus 5 \cdot \left(2^4\right)^3 \oplus 61 \cdot \left(2^4\right)^2 \oplus 81 \cdot \left(2^4\right) \oplus 228 = 213 \cdot 2^{32} \oplus 224 \cdot 2^{28} \oplus$$
$$\oplus 234 \cdot 2^{24} \oplus 229 \cdot 2^{20} \oplus 240 \cdot 2^{16} \oplus 5 \cdot 2^{12} \oplus 61 \cdot 2^8 \oplus 81 \cdot 2^4 \oplus 228 =$$
$$= 213 \cdot 157 \oplus 224 \cdot 24 \oplus 234 \cdot 143 \oplus 229 \cdot 180 \oplus 240 \cdot 76 \oplus 5 \cdot 205 \oplus$$
$$\oplus 61 \cdot 29 \oplus 81 \cdot 16 \oplus 228 = 87 \oplus 245 \oplus 159 \oplus 87 \oplus 226 \oplus 222 \oplus$$
$$\oplus 203 \oplus 121 \oplus 228 = 162 \oplus 159 \oplus 87 \oplus 226 \oplus 222 \oplus 203 \oplus 121 \oplus$$
$$\oplus 228 = 61 \oplus 87 \oplus 226 \oplus 222 \oplus 203 \oplus 121 \oplus 228 = 106 \oplus 226 \oplus$$
$$\oplus 222 \oplus 203 \oplus 121 \oplus 228 = 136 \oplus 222 \oplus 203 \oplus 121 \oplus 228 =$$
$$= 86 \oplus 203 \oplus 121 \oplus 228 = 157 \oplus 121 \oplus 228 = 228 \oplus 228 = 0.$$

Результати проведених обчислень підтверджують теоретичні міркування про те, що для правильної кодової комбінації значення всіх сум, сформованих через поліном синдрому помилки, дорівнюють нулю, а у разі наявності помилки у коді деякі із цих сум обов’язкво не дорівнюють нулю.

2. Обчислимо, з використанням алгоритму Берлекемпа – Мессі, коефіцієнти поліному локаторів помилок.

Оскільки алгоритм Берлекемпа – Мессі є досить складним для розуміння, розглянемо окремо кожну ітерацію.

2.1 Ініціалізація алгоритму: $r = 0$, $\Lambda(x)=1$, $m = -1$, $L = 0$, $B(x)=1$.

2.2 Нульова ітерація, $r = 0$. Відхилення складає:

$$\Delta_0 = \sum_{i=0}^{0} \Lambda_i \cdot S_{0-i+1} = \Lambda_0 \cdot S_1 = 1 \cdot 246 = 246.$$

Оскільки відхилення не є нульовим, перераховуємо локатор помилок, додаючи до нього поліном $B(x)$, помножений на значення Λ_0. Отримуємо наступний результат:

$$\Lambda^*(x) = T(x) = \Lambda(x) \oplus \Delta_0 \cdot B(x) = 246 \cdot x \oplus 1.$$

Тепер, згідно із алгоритмом, необхідно перевірити умову $L < r - m$. Згідно із значеннями параметрів моделі, заданими на нульовій ітерації, $r = 0$, $m = -1$, $L = 0$, тоді, $r - m = 1$, $0 < 1$. Тобто, умова $L < r - m$ виконується. Тому, відповідно до блок-схеми алгоритму Берлекемпа – Мессі, необхідно на поточній ітерації змінити значення L, m та $B(x)$, наступним чином:

$$L^* = r - m = 1; \qquad m = r - L = 0; \qquad L = L^* = 1;$$

$$B(x) = \Delta_0^{-1} \cdot \Lambda(x) = 246^{-1} \cdot 1 = 211 .$$

Остаточно на цій ітерації, згідно із блок-схеми алгоритму, отримуємо наступні значення полінома синдрому помилок $\Lambda(x)$ та полінома модифікації $B(x)$:

$$\Lambda(x) = \Lambda^*(x) = 246 \cdot x \oplus 1; \; B(x) = x \cdot B(x) = 211 \cdot x \oplus 0.$$

Тепер збільшуємо номер ітерації на 1 та перевіряємо умову $r < 2 \cdot t = 4$. Оскільки $r = 1 < 2 \cdot t = 4$, переходимо до наступної ітерації.

2.3 Перша ітерація, $r = 1$. Відхилення складає:

$$\Delta_1 = \sum_{i=0}^{1} \Lambda_i \cdot S_{1-i+1} = (\Lambda_1 \cdot S_1) \oplus (\Lambda_0 \cdot S_2) = 246 \cdot 246 \oplus 1 \cdot 207 = 108.$$

Оскільки відхилення не є нульовим, змінюємо, як і на попередній

ітерації поліном локаторів помилок через обчислення нового поліному $\Lambda^*(x)$:

$$\Lambda^*(x) = \Lambda(x) \oplus \Delta_1 \cdot B(x) = (246 \cdot x \oplus 1) \oplus 108 \cdot (211 \cdot x \oplus 0) = 202 \cdot x \oplus 1.$$

На цій ітерації умова $L < r - m$ не виконується, тому значення L та m залишаємо незмінними. Тоді на першій ітерації значення полінома синдрому помилок $\Lambda(x)$ та полінома модифікації $B(x)$ є наступними:

$$\Lambda(x) = \Lambda^*(x) = 202 \cdot x \oplus 1;\ B(x) = x \cdot B(x) = 211 \cdot x^2 \oplus 0 \cdot x \oplus 0.$$

Після цього збільшуємо номер ітерації r на 1. Оскільки $r = 2 < 2 \cdot t = 4$, переходимо до наступної ітерації.

2.4 Друга ітерація, $r = 2$. Відхилення складає:

$$\Delta_2 = \sum_{i=0}^{2} \Lambda_i \cdot S_{2-i+1} = (\Lambda_2 \cdot S_1) \oplus (\Lambda_1 \cdot S_2) \oplus (\Lambda_0 \cdot S_3) =$$

$$= 0 \cdot 246 \oplus 202 \cdot 207 \oplus 1 \cdot 255 = 160.$$

Оскільки і на цій ітерації відхилення не є нульовим, знову змінюємо поліном локаторів помилок через обчислення нового поліному $\Lambda^*(x)$:

$$\Lambda^*(x) = \Lambda(x) \oplus \Delta_2 \cdot B(x) = (202 \cdot x \oplus 1) \oplus 160 \cdot (211 \cdot x^2 \oplus 0 \cdot x \oplus 0) =$$

$$= 158 \cdot x^2 \oplus 202 \cdot x \oplus 1.$$

На цій ітерації умова $L < r - m$ виконується, тому, згідно з алгоритмом, необхідно змінити значення L та m наступним чином:

$$L^* = r - m = 2; \quad m = r - L = 1; \quad L = L^* = 2,$$

а поліном $B(x)$ переписується у вигляді:

$$B(x) = \Delta_2^{-1} \cdot \Lambda(x) = 160^{-1} \cdot (202 \cdot x \oplus 1) = 45 \cdot x \oplus 28.$$

Остаточно на другій ітерації для поліномів $\Lambda(x)$ та $B(x)$ отримуємо наступні значення:

$$\Lambda(x) = \Lambda^*(x) = 158 \cdot x^2 \oplus 202 \cdot x \oplus 1;\ \ B(x) = x \cdot B(x) = 45 \cdot x^2 \oplus 28 \cdot x \oplus 0.$$

Тепер знову збільшуємо номер ітерації r на 1. Оскільки $r = 3 < 2 \cdot t = 4$, переходимо до наступної ітерації.

2.5 Третя ітерація, $r = 3$. Відхилення складає:

$$\Delta_3 = \sum_{i=0}^{3} \Lambda_i \cdot S_{3-i+1} = (\Lambda_3 \cdot S_1) \oplus (\Lambda_2 \cdot S_2) \oplus (\Lambda_1 \cdot S_3) \oplus (\Lambda_0 \cdot S_4) =$$

$$= 0 \cdot 246 \oplus 158 \cdot 207 \oplus 202 \cdot 255 \oplus 213 = 75 .$$

Відхилення не є нульовим, тому і на цій ітерації обчислюємо значення модифікованого поліному $\Lambda^*(x)$:

$$\Lambda^*(x) = \Lambda(x) \oplus \Delta_3 \cdot B(x) = \left(158 \cdot x^2 \oplus 202 \cdot x \oplus 1\right) \oplus 75 \cdot \left(45 \cdot x^2 \oplus 28 \cdot x \oplus 0\right) =$$

$$= 19 \cdot x^2 \oplus 93 \cdot x \oplus 1 .$$

Умова $L < r - m$ на цій ітерації не виконується, тому, не змінюючи значень L та m, переписуємо поліноми $\Lambda(x)$ та $B(x)$ наступним чином:

$$\Lambda(x) = \Lambda^*(x) = 19 \cdot x^2 \oplus 93 \cdot x \oplus 1 ;$$

$$B(x) = x \cdot B(x) = 45 \cdot x^3 \oplus 28 \cdot x^2 \oplus 0 \cdot x \oplus 0 .$$

Оскільки після збільшення номеру ітерації r на 1 виконується умова $r = 4 = 2 \cdot t$, можна вважати цикл ітерацій закінченим, а значення коефіцієнтів поліному локаторів помилок $\Lambda(x)$ – остаточними. Тобто, для прикладу, який розглядається поліном локаторів помилок записується у вигляді:

$$\Lambda(x) = 19 \cdot x^2 \oplus 93 \cdot x \oplus 1 .$$

Оскільки степінь поліному локаторів помилок $\deg(\Lambda(x)) = 2$ та співпадає із значенням $L = 2$, згідно із алгоритмом Берлекемпа – Мессі цей розв'язок задачі слід вважати правильним.

3. Методом перебирання значень номерів розрядів кодової комбінації визначаємо корені полінома локаторів помилок.

Оскільки код має 9 розрядів, будемо перевіряти лише значення 2^0, 2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}, 2^{-5}, 2^{-6}, 2^{-7} та 2^{-8}. Відповідно, згідно із співвідношенням (3.224), маємо наступні результати:

$2^0 = 1$; $2^{-1} = 2^{(255-1)} = 2^{254} = 142$; $2^{-2} = 2^{(255-2)} = 2^{253} = 71$;

$2^{-3} = 2^{(255-3)} = 2^{252} = 173$; $2^{-4} = 2^{(255-4)} = 2^{251} = 216$;

$2^{-5} = 2^{(255-5)} = 2^{250} = 108;$ $2^{-6} = 2^{(255-6)} = 2^{249} = 54;$

$2^{-7} = 2^{(255-7)} = 2^{248} = 27;$ $2^{-8} = 2^{(255-8)} = 2^{247} = 131.$

Для отриманих значень аргументу x обчислимо значення полінома локаторів помилок $\Lambda(x)$, знайденого раніше.

$$\Lambda\left(2^{0}\right) = \Lambda(1) = 19 \cdot 1^{2} \oplus 93 \cdot 1 \oplus 1 = 19 \oplus 93 \oplus 1 = 79;$$

$$\Lambda\left(2^{-1}\right) = \Lambda(142) = 19 \cdot 142^{2} \oplus 93 \cdot 142 \oplus 1 = 19 \cdot 71 \oplus 160 \oplus 1 =$$
$$= 205 \oplus 161 = 108;$$

$$\Lambda\left(2^{-2}\right) = \Lambda(71) = 19 \cdot 71^{2} \oplus 93 \cdot 71 \oplus 1 = 19 \cdot 216 \oplus 80 \oplus 1 =$$
$$= 116 \oplus 81 = 37;$$

$$\Lambda\left(2^{-3}\right) = \Lambda(173) = 19 \cdot 173^{2} \oplus 93 \cdot 173 \oplus 1 = 19 \cdot 54 \oplus 40 \oplus 1 =$$
$$= 29 \oplus 41 = 52;$$

$$\Lambda\left(2^{-4}\right) = \Lambda(216) = 19 \cdot 216^{2} \oplus 93 \cdot 216 \oplus 1 = 19 \cdot 131 \oplus 20 \oplus 1 =$$
$$= 64 \oplus 21 = 85;$$

$$\Lambda\left(2^{-5}\right) = \Lambda(108) = 19 \cdot 108^{2} \oplus 93 \cdot 108 \oplus 1 = 19 \cdot 233 \oplus 10 \oplus 1 =$$
$$= 16 \oplus 11 = 27;$$

$$\Lambda\left(2^{-6}\right) = \Lambda(54) = 19 \cdot 54^{2} \oplus 93 \cdot 54 \oplus 1 = 19 \cdot 125 \oplus 5 \oplus 1 =$$
$$= 4 \oplus 4 = 0;$$

$$\Lambda\left(2^{-7}\right) = \Lambda(27) = 19 \cdot 27^{2} \oplus 93 \cdot 27 \oplus 1 = 19 \cdot 88 \oplus 140 \oplus 1 =$$
$$= 1 \oplus 141 = 140;$$

$$\Lambda\left(2^{-8}\right) = \Lambda(131) = 19 \cdot 131^{2} \oplus 93 \cdot 131 \oplus 1 = 19 \cdot 22 \oplus 70 \oplus 1 =$$
$$= 71 \oplus 71 = 0.$$

Тобто, згідно із обчисленими значеннями поліному локаторів помилок його коренями є значення $2^{-u_1} = 2^{-8}$ та $2^{-u_2} = 2^{-6}$, звідки можна зробити висновок, що сьомий та дев'ятий розряди кодової комбінації, яка розглядається у цьому прикладі, є помилковими. Як було сказано раніше, у даному випадку розряди нумеруються згідно із степенями поліноміальних складових, тому починається нумерація не з першого, а з нульового розряду, і

проводиться зправа-наліво.

4. Обчислюємо поліном величин помилок $\Omega(x)$.

Згідно із співвідношенням (3.226), за умови відомих поліномів синдромів помилок $S(x)$ та локаторів помилок $\Lambda(x)$, обчислюємо поліном величин помилок $\Omega(x)$ наступним чином:

$$S(x)\cdot\Lambda(x)=\left(213\cdot x^3\oplus 255\cdot x^2\oplus 207\cdot x\oplus 246\right)\cdot\left(19\cdot x^2\oplus 93\cdot x\oplus 1\right)=$$

$$=213\cdot 19\cdot x^5\oplus 255\cdot 19\cdot x^4\oplus 207\cdot 19\cdot x^3\oplus 246\cdot 19\cdot x^2\oplus$$

$$\oplus 213\cdot 93\cdot x^4\oplus 255\cdot 93\cdot x^3\oplus 207\cdot 93\cdot x^2\oplus 246\cdot 93\cdot x\oplus 213\cdot x^3\oplus$$

$$\oplus 255\cdot x^2\oplus 207\cdot x\oplus 246=179\cdot x^5\oplus 87\cdot x^4\oplus 32\cdot x^3\oplus 220\cdot x^2$$

$$\oplus 242\cdot x^4\oplus 245\cdot x^3\oplus 35\cdot x^2\oplus 122\cdot x\oplus 213\cdot x^3\oplus 255\cdot x^2\oplus$$

$$\oplus 207\cdot x\oplus 246=179\cdot x^5\oplus(87\oplus 242)\cdot x^4\oplus(32\oplus 245\oplus 213)\cdot x^3\oplus$$

$$\oplus(220\oplus 35\oplus 255)\cdot x^2\oplus(122\oplus 207)\cdot x\oplus 246=$$

$$=179\cdot x^5\oplus 165\cdot x^4\oplus 0\cdot x^3\oplus 0\cdot x^2\oplus 181\cdot x\oplus 246.$$

Як було відмічено вище, операція взяття остачі за модулем $\left(x^{2t}\right)$ від добутку поліномів $\left(S(x)\cdot\Lambda(x)\right)$ просто свідчить про те, що в отриманому результаті добутку цих поліномів необхідно відкинути складові старших степенів, по четверту включно. Тому для прикладу, який розглядається, поліном величин помилок $\Omega(x)$ остаточно можна записати наступним чином:

$$\Omega(x)=\left(S(x)\cdot\Lambda(x)\right)\bmod\left(x^{2t}\right)=181\cdot x\oplus 246.$$

5. Обчислюємо похідну від полінома локаторів помилок Λ’(x).

Згідно із співвідношенням (3.228) отримуємо наступне значення похідної:

$$\Lambda'(x)=\left(19\cdot x^2\oplus 93\cdot x\oplus 1\right)'=\left(19\cdot x^2\right)'\oplus(93\cdot x)'\oplus(1)'=0\oplus 93\oplus 0=93.$$

Дійсно, похідна від постійної величини $(1)'$ та похідна від квадратичної функції $\left(19\cdot x^2\right)'$, згідно із співвідношеннями (3.228) та (3.231), дорівнюють нулю.

6. Обчислюємо значення помилок.

За умови відомого полінома величин помилок $\Omega(x)$ та похідної від полінома локатора помилок $\Lambda'(x)$, з використанням співвідношення (3.229) обчислюються значення помилок. Для прикладу, який розглядається, отримуємо наступні результати для значень помилок ν_1 та ν_2:

$$\nu_1 = \frac{\Omega\left(2^{-u_1}\right)}{\Lambda'\left(2^{-u_1}\right)} = \frac{181 \cdot 2^{(255-8)} \oplus 246}{93} = \frac{181 \cdot 2^{247} \oplus 246}{93} =$$

$$= \frac{181 \cdot 131 \oplus 246}{93} = \frac{78 \oplus 246}{93} = \frac{184}{93} = 30.$$

$$\nu_2 = \frac{\Omega\left(2^{-u_2}\right)}{\Lambda'\left(2^{-u_2}\right)} = \frac{181 \cdot 2^{(255-6)} \oplus 246}{93} = \frac{181 \cdot 2^{249} \oplus 246}{93} =$$

$$= \frac{181 \cdot 54 \oplus 246}{93} = \frac{37 \oplus 246}{93} = \frac{211}{93} = 6.$$

7. Формуємо поліном, який відповідає вектору помилки *E*(*x*).

Згідно із співвідношенням (3.230) за умови відомих значень помилок ν_1 та ν_2 та помилкових розрядів кодової комбінації u_1 та u_2 вектор помилки $E(x)$ можна записати наступним чином:

$$E(x) = \sum_{i=1}^{2} \nu_l \cdot x^{u_l} = \nu_1 \cdot x^{u_1} \oplus \nu_2 \cdot x^{u_2} = 30 \cdot x^8 \oplus 6 \cdot x^6.$$

8. Формуємо виправлену кодову комбінацію як суму спотвореної комбінації та вектора помилки.

За умови відомого вектора помилки $E(x)$ формування правильної кодової комбінації, згідно із співвідношенням (3.223), зводиться до простої алгебраїчної операції сумування прийнятої спотвореної кодової комбінації із вектором помилки. Результат цієї операції наочно показаний на рис. 3.42. Дійсно, для останнього, дев'ятого розряду прийнятої кодової комбінації, 203 $\oplus$ 30 = 213, для сьомого розряду – 236 $\oplus$ 6 = 234, а інші розряди не змінюються, оскільки для них значення елементів вектора помилки дорівнює 0.

$$
\begin{array}{l}
203\cdot x^8 \oplus 224\cdot x^7 \oplus 236\cdot x^6 \oplus 229\cdot x^5 \oplus 240\cdot x^4 \oplus 5\cdot x^3 \oplus 61\cdot x^2 \oplus 81\cdot x \oplus 228 \\
\oplus \quad 30\cdot x^8 \oplus \quad 0\cdot x^7 \oplus \quad 6\cdot x^6 \oplus \quad 0\cdot x^5 \oplus \quad 0\cdot x^4 \oplus 0\cdot x^3 \oplus \quad 0\cdot x^2 \oplus 81\cdot x \oplus 0 \\
\hline
213\cdot x^8 \oplus 224\cdot x^7 \oplus 234\cdot x^6 \oplus 229\cdot x^5 \oplus 240\cdot x^4 \oplus 5\cdot x^3 \oplus 61\cdot x^2 \oplus 81\cdot x \oplus 228
\end{array}
$$

Рис. 3.42 Результат сумування спотвореної комбінації коду Ріда – Соломона $C(x)$ із визначеним вектором помилки $E(x)$ для прикладу 3.42

Аналіз остаточної отриманої кодової комбінації показує, що вона цілком співпадає із початковою, не спотвореною комбінацію. Тобто, можна зробити висновок, що сформований код Ріда – Соломона працює коректно і дозволяє виправляти подвійні помилки в інформаційних повідомленнях.

Із проведених у цьому прикладі розрахунків стає зрозумілим, що під час реалізації алгоритму декодування кодів Ріда – Соломона досить часто необхідно обчислювати значення поліноміальних функцій від заданого аргументу у визначеному полі Галуа $GF(2^m)$. Оскільки для нас зараз головним поставленим завданням є створення програмного комплексу для декодування кодів Ріда – Соломона, для подальшого спрощення розв'язування цього завдання процедуру для обчислення значень поліноміальних функцій від заданого аргументу варто реалізувати окремим модулем. Програмний код такої процедури **`calcpolgf2m`**, написаний мовою програмування системи MatLab, а також результати її тестування, наведені у додатку Н.

Розглянемо тепер спрощений алгоритм декодування послідовностей кодів Ріда – Соломона, який використовується у разі наявності одиночної помилки [81]. Дійсно, якщо за умовою поставленою задачі заздалегідь відомо, що послідовність кода Ріда – Соломона містить лише одну помилку, спосіб пошуку цієї помилки зводиться до розв'язування у визначеному полі Галуа простого лінійного рівняння. Тобто, використання розглянутого вище складного алгоритму Берлекемпа – Мессі та теореми Форні є зайвим.

У випадку однієї помилки обчислюються лише дві контрольні суми, S_1

та S_2. За такої умови рівняння оптимізації (3.225) значно спрощується та записується у вигляді [81]:

$$j=2; \qquad \Lambda(x)=\left(\frac{S_2}{S_1}\right)\cdot x\oplus 1; \qquad \Lambda_1=\left(\frac{S_2}{S_1}\right). \tag{3.235}$$

Поліном локаторів помилок, заданий другим рівнянням співвідношень (3.235), є лінійною функцією у визначеному полі Галуа $GF(2^m)$. Таке рівняння має єдиний розв'язок u, який, згідно із визначенням 3.12 для від'ємної степені у полі Галуа та із формулами (3.224), (3.235), пов'язаний із контрольними S_1 та S_2 через просте степеневе рівняння [81]:

$$2^{-u}=\left(\frac{S_1}{S_2}\right). \tag{3.236}$$

Отримане рівняння (3.236), враховуючи визначення 3.11 для операції ділення та формулу (3.212), має такий розв'язок [81]:

$$u=\log_2\left(\frac{S_2}{S_1}\right)=\log_2(S_2)-\log_2(S_1). \tag{3.237}$$

За умови відомого поліному локаторів помилок $\Lambda(x)$, заданого другим рівнянням системи (3.235), та поліному синдромів помилок, який у разі виникнення однієї помилки записується у вигляді

$$S(x)=S_2(x)\oplus S_1, \tag{3.238}$$

можна, з використанням теореми Форні та співвідношень (3.226), (3.232), (3.238), знайти аналітичний вираз для поліному величини помилки $\Omega(x)$ [81]:

$$\Omega(x)=(S(x)\cdot\Lambda(x))\bmod(x^2)=\left((S_2(x)\oplus S_1)\cdot\left(\left(\frac{S_2}{S_1}\right)\cdot x\oplus 1\right)\right)\bmod(x^2)=$$

$$=\left((S_2(x)\oplus S_1)\cdot\left(\left(\frac{S_2}{S_1}\right)\cdot x\oplus 1\right)\right)\bmod(x^2)=\left(\left(\frac{S_2^2}{S_1}\right)\cdot x^2\oplus\right.$$

$$\left.\oplus\left(S_2\oplus S_1\cdot\left(\frac{S_2}{S_1}\right)\right)\oplus S_1\right)\bmod(x^2)=$$

$$= \left(\left(\frac{S_2^2}{S_1} \right) \cdot x^2 \oplus (S_2 \oplus S_2) \oplus S_1 \right) \mathrm{mod}\left(x^2\right) =$$

$$= \left(\left(\frac{S_2^2}{S_1} \right) \cdot x^2 \oplus S_1 \right) \mathrm{mod}\left(x^2\right) = S_1 . \qquad (3.239)$$

А похідна від поліному локаторів помилок $\Lambda'(x)$, у разі виникнення одиночної помилки, згідно із співвідношеннями (3.228), (3.231), (3.232) обчислюється наступним чином [81]:

$$\Lambda'(x) = \Lambda_1 = \frac{S_2}{S_1} . \qquad (3.240)$$

Відповідно, для значення помилки, згідно із теоремою Форні та співвідношенням (3.229), (3.239), (3.240), отримуємо наступний вираз [81]:

$$\nu = \frac{\Omega\left(2^{-u}\right)}{\Lambda'\left(2^{-u}\right)} = S_1 \Big/ \left(\frac{S_2}{S_1} \right) = \frac{S_1^2}{S_2} = 2^{\left(2 \cdot \log_2(S_2) - 2 \cdot \log_2(S_1)\right)} . \qquad (3.241)$$

Тут цікавим є те, що, згідно із співвідношенням (3.241), у разі одиночної помилки у коді Ріда – Соломона значення помилки визначається лише контрольними сумами S_1 та S_2 та не залежить від положення помилкового розряду у кодовій комбінації, тобто від значення u. Це безпосередньо пов’язано із принципом формування кодів Ріда – Соломона через твірні поліноми, описаним у підрозділі 3.4.5.

Таким чином, як видно із отриманих співвідношень (3.237) та (3.241), у разі розгляду одиночної помилки у кодах Ріда – Соломона, задачі пошуку її положення у кодовій комбінації та її значення зводяться до елементарних алгебраїчних операцій піднесення до степені та обчислення логарифмів у визначеному полі Галуа $GF(2^m)$, розглянутих у підрозділі 3.4.4.1.

На основі співвідношень (3.237), (3.241), легко формується вектор помилки $E(x)$, заданий векторним рівнянням (3.230). Зрозуміло, що всі розряди цього вектора, за винятком розряду із номером $u + 1$, дорівнюють 0, а розряд із номером $u + 1$ має значення ν. Слід мати на увазі, що номер $u + 1$ відповідає

нумерації розрядів зправа-наліво, як у числовому коді. Значення $u+1$ формується з урахуванням того, що перший розряд у поліноміальному поданні коду завжди вважається нульовим, оскільки він відповідає вільному члену полінома, або степені 0 змінної x. За умови відомого вектора помилки можна поновити передану кодову комбінацію з використанням співвідношення (3.223), тобто, через сумування за модулем два спотвореної комбінації $C(x)$ із обчисленим вектором помилки $E(x)$.

Цікавою є ще одна особливість блокових кодів Ріда – Соломона, пов'язана із способом їхнього формування. Припустимо, що у кодовій послідовності, яка закодована кодом, призначеним для виявлення помилок заданої кратності, виникла лише одна помилка. Виникає питання: що буде, якщо таку послідовність розкодувати не з використанням алгоритму Берлекемпа – Мессі та теореми Форні, тобто, складного ітераційного алгоритму, заданого співвідношеннями (3.220) – (3.231) та наведеного на рис. 3.41, а з використанням спрощених співвідношень (3.237), (3.241)? Тут треба мати на увазі, що контрольні суми, задані співвідношеннями (3.220), (3.221), розраховуються для кодової послідовністю із заданою коректувальною здатністю. А коректувальна здатність залежить від кількості примітивних одночленів $f_i(x)=\left(x \oplus 2^i\right)$, які, згідно із співвідношенням (3.200), перемножуються між собою для формування твірного поліному $g(x)$.

Припустимо, що в результаті ділення модифікованого інформаційного повідомлення на твірний поліном та додавання остачі кодова послідовність систематичного коду Ріда – Соломона має наступний вигляд:

$$[k_n,\ldots, k_2, k_1, r_{2\cdot t}, r_{2\cdot t-1}, \ldots, r_2, r_1], \qquad (3.242)$$

де k – інформаційні розряди, r – контрольні розряди, n – кількість символів в інформаційному повідомленні, t – кількість помилок, які необхідно виявляти та виправляти. Зрозуміло, що контрольні суми S_1 та S_2 для такої послідовності дорівнюють 0, але значення 0 мають і інші контрольні суми, від S_3 до $S_{2\cdot t}$. Якщо

не враховувати цього, а лише аналізувати суми S_1 та S_2 та шукати положення та значення помилки з використанням спрощених співвідношень (3.237), (3.241), одиночна помилка у коді дійсно буде визначена цілком правильно. Проте помилка декодування у цьому випадку полягає у тому, що, згідно із алгоритмом декодування, із вектора кодової послідовності, заданого співвідношенням (3.242), будуть відкинуті не всі контрольні розряди $r_{2\cdot t}$, $r_{2\cdot t-1}$, ..., r_2, r_1, а лише останні два розряди, r_1 та r_2. Тобто, спотворення кодової комбінації за такої умови здійснюється наступним чином:

$$[k_n, \ldots, k_2, k_1, r_{2\cdot t}, r_{2\cdot t-1}, \ldots, r_2, r_1] \rightarrow [k_{n+2\cdot(t-1)}, k_{n+2\cdot(t-2)}, \ldots, k_2, k_1, r_2, r_1].$$

Слід відзначити, що у послідовності (3.242) використана пряма нумерація розрядів коду зправа-наліво, як і для більшості кодових комбінацій у цьому посібнику. Елементи векторів зазвичай нумеруються у зворотному порядку, зліва-направо [13, 14, 48].

Тобто, хоча у разі зменшення параметру максимальної кількості помилок t, які виявляються та виправляються сформованим кодом, алгоритм декодування дозволяє виявляти та виправляти помилки меншої кратності. Хибність такого підходу полягає лише у тому, що втрачається визначена границя між інформаційними та контрольними розрядами та невірно розшифровується закодоване інформаційне слово.

Наприклад, якщо у коді RS = [213, 224, 234, 229, 240, 5, 61, 81, 228], отриманому у прикладі 3.41, шукати не подвійну, а одиночну помилку, закодоване слово C = [213, 224, 234, 229, 240] перетворюється на хибну кодову комбінацію C_x = [213, 224, 234, 229, 240, 5, 61], яка має 2 зайвих розряди. Крім цього, такий код має меншу коректувальну здатність, оскільки він не виявляє та не виправляє подвійні помилки.

З іншого боку, у разі формування поліному локаторів помилок за алгоритмом Берлекемпа – Мессі, який задається оптимізаційним співвідношенням (3.225), степінь поліному завжди є мінімальною та відповідає дійсній кількості помилок у кодовій комбінації τ. Якщо порядок поліному дорівнює 1, можна у будь-якому разі перейти від співвідношень (3.226) – (3.231) до більш простих рівнянь (3.237), (3.241). Різниця полягає у тому, що у

цьому випадку визначаються всі можливі помилки, а границя між інформаційними та контрольними розрядами задається параметром коду *t*.

Тому загальне правило декодування кодових послідовностей для групових коді Ріда – Соломона є наступним.

Правило декодування кодових послідовностей кодів Ріда – Соломона. Максимальна кількість помилок *t*, які виявляються та виправляються у сформованій кодовій комбінації, є незмінним параметром коду, який використовується для пошуку помилок під час декодування. Реальна кількість помилок τ може бути меншою, але вона завжди відповідає степені обчисленого полінома локаторів помилок, $\tau = \deg(\Lambda(x))$.

Розглянемо приклад пошуку одиночної помилки у коді Ріда – Соломона з використанням співвідношень (3.237), (3.241).

Приклад 3.43. Побудувати код Ріда – Соломона, який виправляє одиночну помилку, для інформаційного слова [13, 24, 6, 21] у полі Галуа $GF(2^5)$. Занести помилку у третій другий інформаційного слова та показати, як вона виправляється.

Будемо розв'язувати цю задачу послідовно.

1. Сформуємо твірний поліном у полі Галуа $GF(2^5)$ для виявлення та виправлення одиночної помилки.

$$g(x) = \prod_{i=1}^{2}\left(x \oplus 2^i\right) = \left(x \oplus 2^1\right) \cdot \left(x \oplus 2^2\right) = x^2 + 6 \cdot x + 8.$$

2. Формуємо вектор модифікованої вхідної кодової послідовності [13, 24, 6, 21, 0, 0] та записуємо його у поліноміальній формі:

$$M(x) = 13 \cdot x^5 + 24 \cdot x^4 + 6 \cdot x^3 + 21 \cdot x^2 + 0 \cdot x + 0.$$

3. Ділимо поліном $M(x)$ на твірний поліном $g(x)$ та визначаємо остачу від цього ділення. Результат ділення показаний на рис. 3.43. Оскільки вектором остачі є кодова послідовність r = [3, 8], зрозуміло, що для прикладу, який розглядається, послідовність коду Ріда – Соломона має вигляд: RS = [13, 24, 6, 21, 3, 8].

$$
\begin{array}{l|l}
\oplus\ \begin{array}{l} 13\cdot x^5 + 24\cdot x^4 + 6\cdot x^3 + 21\cdot x^2 + 0\cdot x + 0 \\ 13\cdot x^5 + 11\cdot x^4 + 7\cdot x^3 \end{array} & x^2 + 6\cdot x + 8 \\
\hline
\quad \oplus\ \begin{array}{l} 19\cdot x^4 + 1\cdot x^3 + 21\cdot x^2 \\ 19\cdot x^4 + 5\cdot x^3 + 12\cdot x^2 \end{array} & 13\cdot x^3 + 19\cdot x^2 + 4\cdot x + 1 \\
\hline
\qquad \oplus\ \begin{array}{l} 4\cdot x^3 + 25\cdot x^2 + 0\cdot x \\ 4\cdot x^3 + 24\cdot x^2 + 5\cdot x \end{array} & \\
\hline
\qquad\quad \oplus\ \begin{array}{l} 1\cdot x^2 + 5\cdot x + 0 \\ 1\cdot x^2 + 6\cdot x + 8 \end{array} & \\
\hline
\qquad\qquad 3\cdot x + 8 &
\end{array}
$$

Рис. 3.43 Результат ділення поліномів у полі Галуа $GF(2^5)$ та визначення остачі для прикладу 3.43

4. Заносимо помилку у другий розряд інформаційного слова. Враховуючи те, що кодова комбінація має два додаткових розряди справа, другий розряд інформаційного слова відповідає четвертому розряду отриманої кодової комбінації коду Ріда – Соломона. Припустимо, що спотворена кодова комбінація має вигляд: $RS_{\text{сп}} = [13, 24, 8, 21, 3, 8]$.

5. Обчислюємо першу та другу контрольні суми.

$$
\begin{aligned}
S_1(x_1) = S_1(2) &= 13\cdot 2^5 \oplus 24\cdot 2^4 \oplus 8\cdot 2^3 \oplus 21\cdot 2^2 \oplus 3\cdot 2 \oplus 8 = \\
&= 13\cdot 5 \oplus 24\cdot 16 \oplus 8\cdot 8 \oplus 21\cdot 4 \oplus 6 \oplus 8 = 28 \oplus 25 \oplus 10 \oplus \\
&\oplus 30 \oplus 6 \oplus 8 = 5 \oplus 10 \oplus 30 \oplus 6 \oplus 8 = 15 \oplus 30 \oplus 6 \oplus \\
&\oplus 8 = 17 \oplus 6 \oplus 8 = 23 \oplus 8 = 31.
\end{aligned}
$$

$$
\begin{aligned}
S_2(x_2) = S_2(4) &= 13\cdot 4^5 \oplus 24\cdot 4^4 \oplus 8\cdot 4^3 \oplus 21\cdot 4^2 \oplus 3\cdot 4 \oplus 8 = \\
&= 13\cdot 2^{10} \oplus 24\cdot 2^8 \oplus 8\cdot 2^6 \oplus 21\cdot 2^4 \oplus 3\cdot 4 \oplus 8 = 13\cdot 17 \oplus \\
&\oplus 24\cdot 13 \oplus 8\cdot 10 \oplus 21\cdot 16 \oplus 3\cdot 4 \oplus 8 = 3 \oplus 9 \oplus 26 \oplus \\
&\oplus 23 \oplus 12 \oplus 8 = 10 \oplus 26 \oplus 23 \oplus 12 = 16 \oplus 23 \oplus 12 = \\
&= 7 \oplus 12 \oplus 8 = 11 \oplus 8 = 3.
\end{aligned}
$$

6. Згідно із формулою (3.237) знаходимо положення помилкового розряду у спотвореній кодовій комбінації:

$$
u = \log_2\left(\frac{S_2}{S_1}\right) = \log_2\left(\frac{3}{31}\right) = \log_2(8) = 3,\ 3 + 1 = 4.
$$

Тобто, помилка виникла у четвертому розряді числа. Зрозуміло, що до

обчисленого значення позиції помилки необхідно додати одиницю, оскільки у поліномі нумерація розрядів починається з нульового.

6. Згідно із формулою (3.241) знаходимо значення четвертого елементу вектора помилки. Слід мати на увазі, що цей елемент є четвертим у випадку нумерації розрядів кодової комбінації зправа-наліво, тобто, для звичайного порядку запису чисел. У разі використання зворотного порядку, тобто, нумерації розрядів зліва-направо, яка часто використовується для аналізу векторів, цей елемент у векторі помилки, який містить 7 розрядів, буде третім.

$$\nu = \frac{S_1^2}{S_2} = \frac{31 \cdot 31}{3} = 14.$$

7. Знаходимо значення помилкового, четвертого розряду спотвореної кодової комбінації:

$$8 \oplus 14 = 6.$$

Тобто, одиночна помилка у четвертому розряді спотвореної кодової комбінації виправлена, оскільки знайдене вірне значення помилкового розряду.

Щодо можливостей пошуку багатократних помилок у кодах Ріда – Соломона, то тут найбільш ефективним для визначення положення помилок і сьогодні вважається розглянутий вище алгоритм Берлекемпа – Мессі, проте його суттєвим недоліком є складність розуміння алгебраїчних операцій, які виконуються над поліномами $\Lambda(x)$ та $B(x)$ у полях Галуа. Ці операції були ретельно проаналізовані та обґрунтовані Берлекемпом та Мессі, які довели відповідні математичні теореми теорії кодування. Як було відмічено вище, розв’язок матричної системи рівнянь (3.220) можна шукати будь-яким із методів матричного аналізу. Такі приклади пошуку розв’язків цього рівняння будуть наведені у наступному підрозділі, а у підрозділі 3.6 розглядатимуться спектральні методи декодування групових кодів, які вважаються ще більш ефективними, оскільки в них у значній мірі мінімізована кількість операцій

множення та ділення елементів поля Галуа.

Найбільш важливою перевагою розглянутих у цьому підрозділі групових кодів є те, що, на відміну від лінійних та циклічних двійкових кодів, які розглядалися раніше, групові коди Ріда – Соломона дозволяють виправляти не окремі біти, із яких формуються інформаційні повідомлення, а цілі символи цих повідомлень, які, як відомо, записуються в натуральному коді ASCII, тобто, кодуються байтами [31, 33]. Наприклад, якщо у якомусь символі спотворені не один, а декілька бітів, така помилка з використанням кодів Ріда – Соломона або блокових кодів БЧХ також може бути виявленою та виправленою, і, що вкрай важливо, вона вважається одиночною.

Тому використання у сучасній обчислювальній техніці та у системах обробки інформації групових кодів у значній мірі спрощує завдання формування та кодування інформаційних повідомлень та робить такі системи значно більш надійними та швидкодійними, незважаючи на надлишковість інформаційних повідомлень, які формуються та обробляються або передаються каналами зв’язку.

3.4.7 Пошук коефіцієнтів полінома локаторів помилок через матричні перетворення та через обчислення визначників

Перед вивченням цього підрозділу необхідно повторити четвертий розділ другої частини посібника та підрозділ 3.3.4.2 цієї частини посібника

Пошук розв’язку матричного рівняння (3.220) через матричні перетворення та через обчислення визначників у значній мірі нагадує алгоритм ПГЦ для кодів БЧХ, який було розглянуто у підрозділі 3.3.4.2. Загалом всі матричні методи базуються на тому, що у полі Галуа $GF(2^m)$ виконуються алгебраїчні перетворення над матрицею синдромів помилок $\mathbf{S}$. Найбільш популярними є два способи обробки цієї матриці: метод Гаусса – Зейделя, пов’язаний із сумуванням рядків та із зведенням початкової матриці до еквівалентної діагональної, та через розрахунок визначника матриці

синдромів помилок та обчислення оберненої матриці [13, 14, 48].

Наприклад, у разі аналізу послідовностей кодів Ріда – Соломона, які виправляють подвійні помилки, матричне рівняння (3.220), (3.221) переписується у вигляді:

$$\begin{pmatrix}\Lambda_2\\ \Lambda_1\end{pmatrix}=\begin{pmatrix}S_1 & S_2\\ S_2 & S_3\end{pmatrix}^{-1}\cdot\begin{pmatrix}S_3\\ S_4\end{pmatrix}. \qquad (3.243)$$

Для розрахунку зворотної матриці $\mathbf{S}^{-1}$ через визначник матриці **S** можуть бути використані відомі співвідношення з лінійної алгебри [48]:

$$\mathbf{S}^{-1}=\frac{1}{\det(\mathbf{S})}\cdot\begin{pmatrix}M_{11} & M_{12}\\ M_{21} & M_{22}\end{pmatrix}^{\mathrm{T}};\ \det(\boldsymbol{S})=S_1\cdot S_3\oplus S_2^2, \qquad (3.244)$$

де M_{kl} – мінори матриці синдромів помилок **S**. У полі Галуа $GF(2^m)$ мінори матриці обчислюються, як і у звичайній алгебрі, через алгебраїчні доповнення. Тобто, для обчислення мінору M_{kl} необхідно викреслити із заданої матриці **D** рядок із номером k та стовпчик із номером l, та обчислити визначник отриманої матриці **F** [48]. Єдина відмінність від звичайної алгебри полягає в тому, що у полях Галуа $GF(2^m)$ є зайвим додатковий множник $-1^{(k+l)}$, оскільки тут відсутнє поняття про від'ємні числа. Для матриці другого порядку, якою є матриця синдромів помилок **S**, спосіб обчислення мінорів значно спрощується, оскільки результатами викреслення із неї одного рядка та одного стовпчика є окремі елементи матриці і відпадає необхідність розраховувати їхні визначники. Згідно із наведеними вище міркуваннями, для матриці **S**, заданої першим рівнянням системи (3.244), значення мінорів є наступними:

$$M_{11}=S_3;\quad M_{12}=S_2;\quad M_{21}=S_2;\quad M_{22}=S_1,$$

тобто

$$\mathbf{M}=\begin{pmatrix}M_{11} & M_{12}\\ M_{21} & M_{22}\end{pmatrix}=\begin{pmatrix}S_3 & S_2\\ S_2 & S_1\end{pmatrix}=\mathbf{M}^{\mathrm{T}}. \qquad (3.245)$$

Розглянемо приклад розв'язування матричного рівняння (3.243) з використанням методу Гаусса – Зейделя та співвідношень (3.244), (3.245).

Приклад 3.44. Для значень контрольних сум, отриманих у прикладі

3.42, знайти коефіцієнти поліному локаторів помилок з використанням метода Гаусса – Зейделя та через визначник матриці **S**, з використанням співвідношень (3.244), (3.245).

У прикладі 3.42 були отримані наступні значення контрольних сум: $S_1 = 246$; $S_2 = 207$; $S_3 = 207$; $S_4 = 213$. Тобто, матриця **S**, згідно із співвідношенням (3.243), має наступний вигляд:

$$\mathbf{S} = \begin{pmatrix} S_1 & S_2 \\ S_2 & S_3 \end{pmatrix} = \begin{pmatrix} 246 & 207 \\ 207 & 255 \end{pmatrix}.$$

Знайдемо розв'язок матричного рівняння (3.243) з використанням метода Гаусса – Зейделя. Для цього необхідно звести матрицю **S** до діагональної через множення її рядків на відповідні коефіцієнти та їхнє сумування. Виконаємо відповідні дії над рядками матриці ітераційно, через послідовні кроки.

1. Обчислимо у полі Галуа $GF(2^8)$ коефіцієнт a, на який необхідно помножити перший рядок, щоб у разі сумування його із другим рядком перший елемент другого рядка мав значення 0. Цілком зрозуміло, що цей коефіцієнт складає $a = \frac{S_2}{S_1} = \frac{207}{246} = 202$.

2. Множимо обидва елементи першого рядка у полі Галуа $GF(2^8)$ на обчислене значення a. В результаті маємо:

$$246 \cdot 202 = 207; \qquad 207 \cdot 202 = 95.$$

Тобто, сформований вектор **A** записується у вигляді: **A** = [207, 95].

3. Перший елемент S_3 вектора сум $\mathbf{S_V} = \begin{pmatrix} S_3 \\ S_4 \end{pmatrix}$, який стоїть у правій частині рівняння (3.243), відповідає першому рядку матриці **S**, тому його також необхідно помножити на $a = 202$ у полі Галуа $GF(2^8)$. В результаті маємо:

$$b = 255 \cdot 202 = 27.$$

4. Знаходимо суму за модулем два елементів вектора **A** та другого рядка матриці **S**:

$$\mathbf{C = A} \oplus \mathbf{S_{<2>}} = [207, 95] \oplus [207, 255] = [0, 160].$$

5. Знаходимо суму за модулем два обчисленого елемента b та другого елемента вектора $\mathbf{S_v}$, $\mathbf{S_{v<2>}} = S_4 = 213$.

$$c = b \oplus 213 = 27 \oplus 213 = 206.$$

6. Ділимо отриманий елемент c на другий елемент отриманого вектора $\mathbf{C}$, $\mathbf{C_{<2>}} = 160$. Тут вкрай важливим є те, що перший елемент вектора $\mathbf{C}$, $\mathbf{C_{<1>}}$, має значення 0. Оскільки всі матричні перетворення у полі Галуа $GF(2^8)$ були лінійними, можна сказати, що друге рівняння матричної системи (3.243), яке початково було записано у вигляді $S_2 \cdot \Lambda_2 + S_3 \cdot \Lambda_1 = S_4$, є еквівалентним простому лінійному рівнянню

$$\mathbf{C}_{<\mathbf{2}>} \cdot \Lambda_1 = c. \qquad (3.246)$$

Вкрай важливим результатом проведених розрахунків є те, що отримане лінійне рівняння (3.246) не містить змінної Λ_2, тому воно має у полі Галуа $GF(2^8)$ простий розв'язок відносно змінної Λ_1:

$$\Lambda_1 = \frac{c}{\mathbf{C}_{<\mathbf{2}>}} = \frac{206}{160} = 93,$$

який цілком співпадає із значенням Λ_1, отриманим у прикладі 3.42 з використанням алгоритму Берлекемпа – Мессі.

7. Перше рівняння матричної системи (3.243), яке записується у вигляді $S_2 \cdot \Lambda_2 + S_3 \cdot \Lambda_1 = S_4$, можна переписати в еквівалентному вигляді:

$$\Lambda_2 = \frac{S_4 \oplus S_3 \cdot \Lambda_1}{S_2}. \qquad (3.247)$$

Враховуючи те, що значення Λ_1 є відомим, із співвідношення (3.246) остаточно маємо:

$$\Lambda_2 = \frac{S_4 \oplus S_3 \cdot \Lambda_1}{S_2} = \frac{213 \oplus 255 \cdot 93}{207} = \frac{213 \oplus 245}{207} = \frac{32}{207} = 19.$$

Тобто, обчислені значення коефіцієнтів поліному локаторів помилок Λ_1 та Λ_2 цілком співпадають із результатом, отриманим у прикладі 4.42.

Тепер знайдемо коефіцієнти полінома локаторів помилок Λ_1 та Λ_2

іншим способом, через обчислення визначника матриці синдромів помилок **S**, алгебраїчних доповнень та зворотної матриці $\mathbf{S}^{-1}$ з використанням співвідношень (3.243), (3.244). Для цього необхідно виконати наступні кроки.

1. З використанням другого співвідношення системи рівнянь (3.244) обчислюємо визначник матриці синдромів помилок **S**:

$$\det(S) = S_1 \cdot S_3 \oplus S_2^2 = 246 \cdot 255 \oplus 207 \cdot 207 = 182 \oplus 139 = 61.$$

2. З використанням співвідношення (3.244) обчислюємо матрицю мінорів **M**:

$$\mathbf{M} = \mathbf{M}^{\mathrm{T}} = \begin{pmatrix} S_3 & S_2 \\ S_2 & S_1 \end{pmatrix} = \begin{pmatrix} 255 & 207 \\ 207 & 246 \end{pmatrix}.$$

3. З використанням першого співвідношення системи рівнянь (3.244) обчислюємо матрицю, зворотну до матриці синдромів помилок **S**:

$$\mathbf{S}^{-1} = \frac{1}{\det(\mathbf{S})} \cdot \mathbf{M}^{\mathrm{T}} = \frac{1}{61} \cdot \begin{pmatrix} 255 & 207 \\ 207 & 246 \end{pmatrix} = 12 \cdot \begin{pmatrix} 255 & 207 \\ 207 & 246 \end{pmatrix} = \begin{pmatrix} 112 & 45 \\ 45 & 28 \end{pmatrix}.$$

4. З використанням матричного рівняння (3.243) обчислюємо вектор коефіцієнтів поліному локаторів помилок Λ.

$$\begin{pmatrix} \Lambda_2 \\ \Lambda_1 \end{pmatrix} = \mathbf{S}^{-1} \cdot \begin{pmatrix} S_3 \\ S_4 \end{pmatrix} = \begin{pmatrix} 112 & 45 \\ 45 & 28 \end{pmatrix} \cdot \begin{pmatrix} 255 \\ 213 \end{pmatrix} =$$

$$= \begin{pmatrix} 112 \cdot 255 \oplus 45 \cdot 213 \\ 45 \cdot 255 \oplus 28 \cdot 213 \end{pmatrix} = \begin{pmatrix} 236 \oplus 255 \\ 25 \oplus 68 \end{pmatrix} = \begin{pmatrix} 19 \\ 93 \end{pmatrix}.$$

Тобто, отримуємо такі ж самі значення коефіцієнтів поліному локаторів помилок, як і у прикладі 4.42: $\Lambda_1 = 93$ та $\Lambda_2 = 19$.

3.4.8 Порівняльний аналіз алгоритму Берлекемпа – Мессі з матричними алгоритмами та обрання способу написання програми, призначеної для декодування послідовностей кодів Ріда – Соломона

Після розгляду прикладів 4.42 та 4.44, у яких пошук коефіцієнтів поліному локаторів помилок для послідовностей кодів Ріда – Соломона здійснювався двома способами, з використанням алгоритму Берлекемпа – Мессі та методів матричного аналізу, проведемо порівняльний аналіз використаних методів. Головною метою цього аналізу є обрання найбільш

ефективного методу для реалізації у комп'ютерній програмі, призначеній для пошуку та виправлення помилок у спотворених послідовностях кодів Ріда – Соломона. Проводячи такий аналіз, будемо виходити із наступних чотирьох критеріїв.

1. Простота та наочність алгоритму.

2. Наявність програмних засобів для реалізації цього алгоритму.

3. Швидкість проведення розрахунків з урахуванням часу виконання елементарних алгебраїчних операцій обраного алгоритму та необхідної кількості ітерацій.

4. Функціональні можливості алгоритму. Тут головним критерієм є можливість виявлення у кодовій послідовності помилок, кількість яких є меншою за максимальне значення t для визначеного типу коду.

Відразу слід відзначити, що головним недоліком алгоритму Берлекемпа – Мессі, описаного у підрозділах 3.3.4.3 та 3.4.6, є складність розуміння його головних кроків. У підрозділі 3.4.6 підкреслювалось, що головною ідеєю цього алгоритму є обчислення на кожному кроці розбіжності між поліномом локаторів помилок $\Lambda(x)$ та лінійними рівняннями системи (3.202). Всі рівняння розглядаються послідовно, і якщо розбіжність існує, до створеного поліному $\Lambda(x)$ додається поліном модифікації $B(x)$. Саме таким чином розв'язується рівняння оптимізації (3.225) для коефіцієнтів поліному локаторів помилок $\Lambda(x)$. Метод пошуку поліномів модифікації $B(x)$ є наслідком складних теорем теорії кодування, тому сутність обчислень за алгоритмом Берлекемпа – Мессі не є простою та його важко відразу зрозуміти. Крім цього, незважаючи на велику кількість написаних програм для реалізації алгебраїчних операцій у полях Галуа $GF(2^m)$, алгоритм Берлекемпа – Мессі ще не був реалізований.

На перший погляд здається, що найбільше переваг має метод Гаусса – Зейделя. Він є простим та зрозумілим та для його реалізації можуть бути використані функції додавання, множення та ділення елементів поля Галуа, наведені у додатку Н, а також функції для здійснення відповідних операцій

над векторами. Проте слід мати на увазі, що цей алгоритм потребує більшої кількості алгебраїчних операцій у полі Галуа $GF(2^m)$, і це, насамперед, операції множення та ділення, які у полі Галуа виконуються через обчислення логарифмів. А логарифми обчислюються через взяття степенів та пошук співпадань, тобто, через комбінаторне перебирання значень степені. Кількість операцій перебирання, які необхідно виконати, зростає степенево відносно порядку поля Галуа m, тому для полів Галуа високого порядку обчислювальна складність метода Гаусса – Зейделя стає непомірно високою.

Щодо алгоритму Берлекемпа – Мессі, він має мінімальну кількість множень елементів у полі Галуа, а виконується він завжди за $2 \cdot t$ кроків, де t – максимальна кількість помилок, які виправляються. Крім того, в цьому алгоритмі, блок-схема якого наведена на рис. 3.8, найскладнішими алгебраїчними операціями у полі Галуа $GF(2^m)$ є множення поліному на число та сумування поліномів, операції множення та ділення поліномів в ньому не використовується.

Проте найважливішою перевагою алгоритму Берлекемпа – Мессі є його вкрай висока функціональність. У разі, якщо кількість помилок у кодовій комбінації є меншою за максимальну, просто формується поліном локаторів помилок відповідного порядку і всі помилки виправляються з використанням співвідношень (3.220) – (3.230). Щодо матричних методів, то у разі наявності меншої кількості помилок, ніж максимальна, матриця **S** є виродженою і матричне рівняння (3.221) не може бути розв’язаним. У цьому випадку для коректного розв’язування задачі декодування кодової послідовності необхідно зменшувати розмірність матриці **S** та проводити обчислення для матриці меншого порядку. Ця особливість обчислювального алгоритму у значній мірі знижує ефективність використання методу Гаусса – Зейделя у комп’ютерних програмах, особливо за умови використання кодів із високою коректувальною здатністю.

Щодо методу, пов’язаному із розрахунком визначника матриці **S**, він є простим лише для матриць другого та третього порядків, а для матриць більш

високих порядків його обчислювальна складність непомірно зростає пропорційно кубу розмірності матриці n^3, і вже для середніх значень n навіть перевищує складність методу Гаусса – Зейделя [48]. Обчислення визначників також ґрунтується на алгебраїчних операціях множення та ділення елементів поля Галуа, які є нібито простими для розуміння, але складним з обчислювальної точки зору. Крім цього, функції обчислення визначників також ще не були реалізовані, їх необхідно створювати окремо.

Проведений аналіз ефективності розглянутих алгоритмів показує, що для програмної реалізації найбільш ефективним є алгоритм Берлекемпа – Мессі, а матричні методи є зручними лише для ручних розрахунків та для кодових комбінацій із невисокою коректувальною здатністю. Результати проведеного вище порівняльного аналізу розглянутих алгоритмів зведені у таблиці 3.18.

Таблиця 3.18 – Порівняльний аналіз алгоритму Берлекемпа – Мессі та матричних алгоритмів

Тип алгоритму	Критерії ефективності			
	Простота	Наявність програмних засобів	Швидкість проведення розрахунків	Функціональні можливості
Алгоритм Берлекемпа – Мессі	Складний	Немає	Дуже висока	Дуже високі
Метод Гаусса – Зейделя	Простий	Існують	Низька для матриць високих порядків	Низькі
Обчислення визначників	Не дуже складний	Частково існують	Низька для матриць високих	Низькі

			порядків	

3.4.9 Описання особливостей реалізації програми, призначеної для формування та декодування послідовностей кодів Ріда – Соломона

Тобто, для написання програми декодування послідовностей кодів Ріда – Соломона та пошуку помилок у них оберемо алгоритм Берлекемпа – Мессі та теорему Форні, які були розглянуті у підрозділі 3.4.6. У цьому разі найбільш складними алгебраїчними операціями з обчислювальної точки зору є наступні.

1. Обчислення контрольних сум.

2. Пошук коренів поліному локаторів помилок прямим перебиранням значень розрядів коду з використанням процедури Ченя.

3. Пошук значень вектору помилок з використанням теореми Форні.

Всі ці операції потребують обчислення значень поліномів високих порядків у полі Галуа $GF(2^m)$, тобто, вони пов'язані із виконанням великої кількості елементарних операцій множення та, як наслідок, із комбінаторним перебором значень степеневої функції.

Комп'ютерна програма **RSC**, яка виконує операцію декодування послідовностей кодів Ріда – Соломона з використанням алгоритму Берлекемпа – Мессі та теореми Форні, наведена у додатку О. Ця програма, як і інші програми, коди яких наведені у цьому посібнику, написана мовою програмування системи науково-технічних розрахунків MatLab. Головними особливостями цієї програми є наступні.

1. Програма **RSC** є багатофункціональною та може бути використаною як для формування кодів Ріда – Соломона, так і для декодування їхніх кодових послідовностей. Тип операції, яку виконує програма, визначається вхідним

параметром **nop**. За умови **nop = 1** виконується операція кодування, а за умови **nop = 2** – операція декодування, будь-які інші значення цього вхідного параметру програми є неможливими. Слід відзначити, що аналогічним чином цей параметр використовується в інших програмах цього посібника, призначених для формування завадостійких кодів та їхнього декодування.

2. Іншими вхідними параметрами програми **RSC** є наступні:

vin – вхідний вектор з елементами поля Галуа $GF(2^m)$. Залежно від типу операції, заданого параметром **nop**, цей вектор визначає або початкову числову послідовність, яку необхідно закодувати, або сформовану послідовність коду Ріда – Соломона, яку необхідно декодувати.

m – порядок поля Галуа. Цей параметр може приймати значення 3, 4, 5, 6, 7, 8, 12 або 16.

ne – максимальна кількість помилок, які виправляються даним типом коду Ріда – Соломона. Цей параметр може приймати значення від 1 до 6.

3. Вихідним параметром програми **RSC** є вектор з елементами поля Галуа $GF(2^m)$, який, у разі здійснення операції кодування, визначає сформований код Ріда – Соломона, а у разі проведення операції декодування – закодоване інформаційне слово. Якщо проводиться операція декодування і знайти правильну кодову послідовність неможливо – програма повертає значення порожнього вектора [ ], який не містить жодного елементу.

4. Програма **RSC** написана за модульним принципом. Це виявляється у тому, що елементарні алгебраїчні операції над елементами полів Галуа $GF(2^m)$ та над поліномами, які формуються з використанням цих елементів, здійснюються через виклик відповідних функцій, які були описані у підрозділах 3.4.4.1 та 3.4.4.2 та наведені у додатку Н. Нижче наведений повний список цих функцій, параметри їхнього виклику, а також описане їхнє призначення.

conv2bin (i, m) – функція для перетворення елементів **i**, які належать полю Галуа $GF(2^m)$, до двійкових послідовностей, які їм відповідають. Параметром функції **i** може бути число або вектор. Результат роботи цієї функції – матриця із кількістю рядків **m** та із кількістю стовпчиків **n = length(i)**, де **n** – кількість елементів вхідного вектора **i**. Двійкові коди елементів для заданої векторної послідовності записуються у стовпчики матриці, яка формується.

convbin2dec (i) – функція для перетворення векторів двійкових кодових послідовностей до відповідних десяткових чисел.

sumgf2m(m, i, j) – функція для сумування елементів **i** та **j**, які належать полю Галуа $GF(2^m)$. Слід відзначити, що вхідними параметрами цієї функції також можуть бути вектора однакової розмірності, у цьому випадку результатом сумування є вектор тієї самої розмірності.

powgf2m(m, i) – степенева функція. Ця функція повертає значення 2^i для елемента **i**, який належить полю Галуа $GF(2^m)$.

loggf2m(m, i) – логарифмічна функція. Ця функція обчислюється як зворотна до степеневої, тобто, якщо $c = 2^i$, тоді $i = \log_2(c)$.

prodgf2m(m, i, j) – функція для множення елементів **i** та **j**, які належать полю Галуа $GF(2^m)$. Слід відзначити, що першим множником **i** може бути вектор, у цьому випадку результатом роботи цієї функції буде поліноміальне множення елементів вектора **i** на число **j**.

divgf2m(m, i, j) – функція для ділення елементів **i** та **j**, які належать полю Галуа $GF(2^m)$ **i** – ділене, **j** – дільник. Слід відзначити, що діленим може бути вектор, у цьому випадку результатом роботи цієї функції буде поліноміальне ділення елементів вектора **i** на число **j**.

prodpolgf2m(m,vin1,vin2)– функція для множення поліномів, заданих векторами **vin1** та **vin2** у полі Галуа $GF(2^m)$.

divpolgf2m(m,vin1,vin2) – функція для ділення поліномів, заданих векторами **vin1** та **vin2** у полі Галуа $GF(2^m)$.

calcpolgf2m(m,vin,k) – функція для обчислення у полі Галуа $GF(2^m)$ значення поліноміальної функції, заданої вектором **vin**, для заданого значення аргументу **k**.

Тоді для формування кодів Ріда – Соломона може бути використаний алгоритм, блок-схема якого наведена на рис. 3.36, а для їхнього декодування – алгоритм, блок-схема якого наведена на рис. 3.41. Розглянемо окремо кроки цих алгоритмів для процедур кодування та декодування з точки зору способів їхньої програмної реалізації та використання відповідних функцій, призначених для здійснення алгебраїчних операцій у полях Галуа $GF(2^m)$.

Для процесу кодування це буде наступна послідовність операцій.

1. Через множення примітивних поліномів $f(x)=\left(x \oplus 2^i\right)$ формується твірний поліном $g(x)$, заданий співвідношенням (3.200). Для здійснення цієї операції використовується функція **prodpolgf2m**. Твірний поліном для формування кодів Ріда – Соломона у програмі позначений ім'ям змінної **TPRS**.

2. До вхідної кодової комбінації, заданої вектором **vin**, дописується **2*ne** нулів, де **ne** – наперед задана кількість помилок, які має виправляти код. В результаті створюється модифіковане кодове слово, яке у програмі позначене ім'ям змінної **VDIV**.

3. Модифіковане кодове слово **VDIV**, з використанням функції **divpolgf2m**, ділиться на твірний поліном **TPRS**, та визначається остача від цього ділення **REST**.

4. До молодших розрядів модифікованого кодового слова вписується обчислення значення остачі **REST**. Отримана кодова послідовність, згідно із блок-схемою алгоритму, наведеною на рис. 3.36, є кодом Ріда – Соломона.

Для здійснення процесу кодування послідовність операцій є наступною.

1. Для введеної кодової послідовності **vin**, за умови заданої максимальної

кількості помилок **ne**, які виправляються кодом, обчислюються контрольні суми **S**.

Слід відзначити, що саме обчислення контрольних сум є найбільш ресурсоємним процесом для розв'язування завдання декодування послідовностей кодів Ріда – Соломона, оскільки ця задача потребує виконання великої кількості алгебраїчних операцій множення чисел у полі Галуа $GF(2^m)$. В принципі, для обчислення значень поліноміальної функції, заданої вхідним вектором **vin**, за умови відомих значень аргументів 2^i може бути використана розглянута вище функція **calcpolgf2m**, яка призначена саме для роботи з поліномами. Проте у програмі **RSC** ця операція виконана трішки інакше, а саме – через створення двомірного масиву **c(i,k)**. Спосіб формування цього масиву числових даних можна легко зрозуміти, якщо безпосередньо проаналізувати метод формування контрольних сум.

Позначимо розмірність вхідного вектора **vin** змінною **nv**. Для обчислення значення поліному, заданого цим вектором, від значень аргументу 2^i, $i = 1...2\cdot$**ne**, необхідно знайти значення степеневої функції від двох аргументів $f(i,k) = \left(2^i\right)^k$, **k=1...nv-1**, та помножити знайдені величини $f(i,k)$ на компоненти вхідного вектора **vin(nv-k)**. Деяка плутанина в індексації елементів вектора через показники степені обумовлена використанням прямого порядку, зправа-наліво, для нумерації розрядів у поліноміальному поданні кодової послідовності, та зворотного порядку, зліва-направо, для нумерації елементів вектора. Взагалі проблема використання прямого та зворотного порядку запису та читання розрядів числа є однією із головних для забезпечення коректного функціонування кодувальних та декодувальних програмних засобів та відповідних цифрових електронних схем. Простий спосіб переведення числових кодових послідовностей з одного порядку запису розрядів до іншого з використанням засобів програмування системи MatLab було розглянуто у підрозділі 1.3.4.

З урахуванням вище сказаного, побудуємо матрицю значень степеневої

функції $c(i,k)=f(i,k)$ у полі Галуа $GF(2^m)$ наступним чином. У перший рядок запишемо значення 2^i, у другий – $2^{2\cdot i}$, і так далі. Тоді у довільному рядку k стоятимуть значення $2^{k\cdot i}$, а в останньому рядку із номером **nv** – значення $2^{\mathbf{nv}\cdot i}$. Наочно спосіб формування матриці $c(i,k)$ показаний на рис. 3.44.

Тоді, з використанням сформованої матриці значень степеневої функції **c** можна записати наступний вираз для обчислення синдромів помилок:

$$S_i = \sum_{j=1}^{\mathrm{nv}-1} \mathrm{vin}_j \cdot c_{\mathrm{nv}-j,i} \oplus c_{\mathrm{nv}},\ i = 1...2\cdot\mathbf{ne}, \qquad (3.247)$$

або у матричній формі:

$$\mathbf{S} = \mathbf{c_m}\cdot\mathbf{vin_{inv}}, \qquad (3.248)$$

де – $\mathbf{c_m}$ – модифікована матриця, яка створюється із матриці **c** шляхом додавання зліва стовпчика із одиниць, які відповідають значенню 2^0, $\mathbf{vin_{inv}}$ – інверсний вектор вхідної кодової послідовності, у якому розряди кодової комбінації записані у зворотному порядку, зліва-направо.

$$\mathbf{c} = \begin{pmatrix} c_{1,1}=2^1; & c_{1,2}=2^2; & \dots & c_{1,i}=2^i; & \dots & c_{1,2\cdot\mathrm{ne}}=2^{2\cdot\mathrm{ne}}; \\ c_{2,1}=2^2; & c_{2,2}=\left(2^2\right)^2=2^4; & \dots & c_{2,i}=\left(2^i\right)^2=2^{2\cdot i}; & \dots & c_{2,2\cdot\mathrm{ne}}=\left(2^{2\cdot\mathrm{ne}}\right)^2=2^{4\cdot\mathrm{ne}}; \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{k,1}=2^k; & c_{k,2}=\left(2^2\right)^k=2^{2\cdot k}; & \dots & c_{k,i}=\left(2^i\right)^k=2^{i\cdot k}; & \dots & c_{k,2\cdot\mathrm{ne}}=\left(2^{2\cdot\mathrm{ne}}\right)^k=2^{2\cdot k\cdot\mathrm{ne}}; \\ \dots & \dots & \dots & \dots & \dots & \dots \\ c_{\mathrm{nv}-1,1}=2^{\mathrm{nv}-1}; & c_{\mathrm{nv}-1,2}=\left(2^2\right)^{\mathrm{nv}-1}=2^{2\cdot(\mathrm{nv}-1)}; & \dots & c_{\mathrm{nv}-1,k}=\left(2^k\right)^{\mathrm{nv}-1}=2^{k\cdot(\mathrm{nv}-1)}; & \dots & c_{\mathrm{nv}-1,2\cdot\mathrm{ne}}=\left(2^{2\cdot\mathrm{ne}}\right)^{\mathrm{nv}-1}=2^{2\cdot\mathrm{ne}\cdot(\mathrm{nv}-1)} \end{pmatrix}.$$

Рис. 3.44 Наочна ілюстрація способу обчислення значень степеневої функції від двох аргументів $c(i,k)=f(i,k)$ у полі Галуа $GF(2^m)$

Для проведення розрахунків компонентів вектору синдромів помилок **S** у програмі **RSC** безпосередньо використовується співвідношення (3.247).

2. Якщо всі контрольні суми S_i для значень $i = 1...2\cdot\mathbf{ne}$ дорівнюють нулю, вважається, що кодова комбінація є правильною процес та декодування закінчується. Для отримання закодованого інформаційного слова достатньо видалити із вхідного вектора **vin** $2\cdot\mathbf{ne}$ останніх розрядів. Тобто, видаляються праві, молодші розряди заданої кодової комбінації.

3. Якщо хоча б одна із обчислених контрольних сум S_i не дорівнює нулю, з використанням алгоритму Берлекемпа – Мессі, блок-схема якого наведена на рис. 3.8, шукаються коефіцієнти поліному локаторів помилок. Вектор цих коефіцієнтів у програмі **RSC** позначений ім'ям змінної **LBD_POL**. Зрозуміло, що реалізація алгоритму Берлекемпа – Мессі є найскладнішою частиною програми декодування кодів Ріда – Соломона, тому ця частина програми нижче буде розглянута окремо.

4. До сформованого поліному локаторів помилок підставляються значення $j = 2^{-ii}$, де **ii** – номери розрядів сформованої кодової комбінації. Для здійснення цієї алгебраїчної операції використовується описана вище функція **calcpolgf2m**. У програмі значенням $\Lambda(j)$ відповідає змінна **cc**. На початку процедури визначення положення помилок вводиться допоміжна змінна номеру помилки **jj**.

5. За умови **cc = 0** розряд кодової комбінації вважається помилковим. Тоді значення **jj** збільшується на 1 та фіксується номер помилкового розряду **u**(**jj**) = **ii**. У такий спосіб формується вектор номерів помилкових розрядів **u**.

6. Якщо для всіх розрядів заданої кодової комбінації **vin** умова **cc = 0** не виконується, вважається, що така комбінація містить занадто велику кількість помилок, які не можливо виправити, та обчислювальний процес примусово припиняється.

7. Формується поліном значень помилок $\Omega(x)$. Для цього, з використанням функції **prodpolgf2m**, сформований поліном локаторів помилок $\Lambda(x)$ множиться на поліном синдромів помилок $S(x)$, та із отриманого результату $\Omega^*(x) = \Lambda(x) \cdot S(x)$ видаляються старші розряди кодової комбінації, степінь яких перевищує величину **2*ne**. У програмі **RSC** коефіцієнтам поліному $\Omega(x)$ відповідає вектор **EVP**.

8. З використанням співвідношень (3.231) формується аналітичний вираз

для похідної від поліному локаторів помилок $\Lambda'(x)$.

9. З використанням співвідношення (3.229) обчислюються значення помилок, на основі яких, з використанням співвідношення (3.230), формується вектор помилки $E(x)$. Цьому вектору у програмі **RSC** відповідає змінна з ім'ям **Err_Vect**. Для обчислення значень помилок використовуються описані вище функції **powgf2m**, **calcpolgf2m** та **divgf2m**.

10. Обчислений вектор помилок **Err_Vect** сумується за модулем два із вхідним вектором **vin**, в результаті чого формується нова кодова комбінація із виправленими помилками. В кінці роботи програми **RSC** на екран видається повідомлення про позиції помилкових розрядів та їхні значення, а також повертається до головної програми вектор закодованого інформаційного слова **vout**.

Розглянемо тепер більш досконало особливості реалізації у програмі **RSC** алгоритму Берлекемпа – Мессі, блок-схема якого наведена на рис. 3.8.

Для проведення обчислень за алгоритмом Берлекемпа – Мессі у програмі **RSC** використані наступні змінні.

ri – номер ітерації.

DELT – відхилення значень коефіцієнтів поліному локаторів помилок від відповідного рівняння.

DELT_COMP – значення відхилення, розраховане на поточній ітерації.

DELT_INV – значення, зворотне до величини відхилення **DELT**, яке обчислюється у поля Галуа $GF(2^m)$.

mi – номер ітерації, на якій останній раз змінювались коефіцієнти поліному локаторів помилок.

L – розрахована кількість спотворених байтів.

NLBD_POL – розмірність обчисленого поліному локаторів помилок.

LBD_POL_n – нове значення поліному локаторів помилок.

B – поліном модифікації $B(x)$.

POLB – результат множення відхилення **DELT** на поліном $B(x)$.

Для виконання алгебраїчних операцій додавання, множення та ділення у полі Галуа $GF(2^m)$ у програмі **RSC** використані описані вище функції **sumgf2m**, **prodgf2m** та **divgf2m**. Обчислення за алгоритмом Берлекемпа – Мессі реалізовані ітераційно з використанням оператору циклу із передумовою **while** [13, 14]. Умова виходу із циклу лише одна: загальна кількість ітерацій **ri** не повинна перевищувати величини **2*ne**.

Цікавим є також дослідження часу виконання програми **RSC**, який необхідний для розв'язування завдань формування та декодування різних кодових послідовностей. Відповідні тести були проведені на персональному комп'ютері з процесором Intel Celeron, тактова частота – 1,1 МГц, об'єм оперативної пам'яті – 512 Мб. Для здійснення тестів була написана програма **TESTRSC**, код якої та результати її роботи також наведений у додатку О. У програмі **TESTRSC** для оцінки часу роботи програми **RSC** у разі формування та декодування різних послідовностей кодів Ріда – Соломона використана системна функція MatLab **cputime**. Як відомо з літературних джерел, ця функція повертає умовний час роботи числового процесора системи MatLab після її останнього завантаження, виражений у тактових інтервалах [13, 14]. Тому, якщо за допомогою цієї функції фіксувати час роботи системи перед завантаженням програми **RSC** та після завершення її роботи, обчислена різниця цих двох величин дає приблизну відносну оцінку часового інтервалу, використаного комп'ютером для розв'язування відповідного завдання.

Аналіз результатів тестування програми **RSC**, наведених у додатку О, дозволяє зробити наступні висновки.

По-перш за все слід відзначити, що завдання створення послідовностей кодів Ріда – Соломона потребує дещо менших витрат машинного часу, ніж завдання декодування. Насамперед це пов'язано з тим, що алгебраїчна

операція множення поліномів першого порядку для створення твірного поліному потребує використання меншої кількості операцій множення елементів поля Галуа, ніж обчислення значень поліному синдромів помилок для різних значень аргументу 2^i [52, 62, 63]. Наприклад, формування коду Ріда – Соломона, який виправляє одну помилку, для послідовності $\mathbf{C} = [13,15,12,11,13]$ у полі Галуа $GF(2^4)$ здійснюється за 0,32 умовних тактів, а декодування сформованої послідовності та пошук помилки в ній потребує 0,9 умовних тактів.

Час роботи алгоритму декодування пропорційний загальній довжині кодової послідовності n. Крім цього, завдання пошуку помилок подвійної та більшої кратності є більш ресурсоємним з точки зору використання машинного часу, ніж пошук одиночної помилки. Наприклад, для визначеної вхідної кодової послідовності $\mathbf{C} = [13,15,12,11,13]$ пошук одиночної помилки здійснюється за 0,9 процесорних тактів, а потрійної – за 1,8 процесорних тактів. Слід відзначити, що збільшення довжини кодової послідовності n може бути обумовлено двома вагомими причинами: збільшенням кількості розрядів у інформаційному слові, яке кодується, та зростанням величини максимальної кількості помилок, які виявляються та виправляються кодом.

Проте проведені тести показали, що час розв'язування завдань кодування та декодування значно більше залежить від порядку поля Галуа m, ніж від довжини кодової послідовності n. Якщо залежність $t(n)$ є лінійною, то залежність $t(m)$ – степеневою. Така закономірність пов'язана із тим, що алгебраїчна операція множення елементів поля Галуа потребує комбінаторного перебирання елементів поля для обчислення значень логарифмів, а кількість елементів поля пропорційна 2^m. Наприклад, код Ріда – Соломона із виправленням однієї помилки для поля Галуа $GF(2^4)$ створюється за 0,32 умовних процесорних тактів, для поля Галуа $GF(2^8)$ – за 5,6 процесорних тактів, а для поля Галуа $GF(2^{16})$ – за 815,7 процесорних тактів. Декодування кодової послідовності у полі Галуа $GF(2^{16})$ із виявленням однієї помилки здійснюється за 1273,3 процесорних тактів. Час формування кодової

послідовності із можливістю виправлення однієї помилки для інформаційного слова $\mathbf{C} = [2341, 3433, 1865]$ у полі Галуа $GF(2^{12})$ становить приблизно 91 процесорний такт, а пошук помилки у сформованому коді здійснюється приблизно за 183,7 процесорних тактів.

Зрозуміло, що найбільш ресурсоємним завданням з точки зору використання машинного часу є обчислення контрольних сум, а найменш ресурсоємним – розрахунки за алгоритмом Берлекемпа – Мессі. Щодо обчислення значень поліному локаторів помилок для пошуку номерів помилкових розрядів, ця задача не є складною з обчислювальної точки зору, оскільки обчислення проводяться не для всіх елементів поля Галуа, а лише для чисел від 1 до n, де n – довжина кодової комбінації. Крім цього, поліном локаторів помилок зазвичай має менший порядок, ніж поліном, заданий вхідною кодовою послідовністю. Найбільший можливий порядок поліному локаторів помилок $\deg_{max}(\Lambda(x))$ відповідає максимальному значенню кількості помилок **ne**.

З іншого боку, використання процедури Форні потребує обчислення значень поліному, значень помилок та похідної від поліному локаторів помилок, а також неодноразового виконання операції ділення елементів поля Галуа, вкрай ресурсоємної з обчислювальної точки зору. Обчислювальна задача пошуку значень помилок у групових кодах дещо спрощується у разі використання спектральних методів їхнього формування та декодування, які розглядатимуться у підрозділі 3.6. Але виконання найбільш ресурсоємної задачі, пов'язаної із обчислення значень поліному синдромів помилок, у будь-якому разі уникнути не вдається [63].

3.4.10 Двійкові коди Ріда – Соломона та особливості їхнього формування та декодування

Перед вивченням цього підрозділу необхідно повторити другий та третій розділи другої частини посібника

Зрозуміло, що теоретичний матеріал, призначений описанню способів формування кодів Ріда – Соломона та декодування їхніх послідовностей, який

наведений у підрозділах 3.4.5, 3.4.6 та 3.4.7, є досить загальним. Описані методи та алгоритми є універсальними та можуть бути використані як для двійкових, так і для багаторозрядних кодів. Проте формування кодів Ріда – Соломона для двійкових послідовностей має свої певні особливості, які слід розглянути окремо [33, 52, 62].

Зазвичай відповідні методи кодування та декодування двійкових послідовностей з використанням кодів Ріда – Соломона базуються на процедурі відображення багатопозиційних кодів Ріда – Соломона на двійкові коди, яка була розглянута у підрозділі 3.4.3.

Слід також відзначити, що процес формування кодів Ріда – Соломона для двійкових послідовностей зазвичай здійснюється не через алгебраїчні операції множення та ділення елементів поля, а через степені примітивного елемента α. Відповідний математичний апарат був розглянутий у другому та третьому розділах другої частини посібника [48].

Хоча, з одного боку, операції додавання та множення у полях Галуа $GF(2^m)$ цілком коректно визначаються операцією «виключного або» та визначенням 3.10, їх можна переписати і через степені примітивного елемента групи α. Наприклад, для поля Галуа $GF(2^3)$ таблиці додавання та множення, записані через примітивний елемент α, відповідають таблицям 3.19 та 3.20 [33]. Неважко зрозуміти, що оскільки розглядається двійковий код, α = 2.

Значення степенів примітивного елемента α у двійковій формі наведені у таблиці 3.21 [33]. Цілком зрозуміло, що ці значення відповідають десятковим значенням, наведеним у таблиці 3.16. Проте слід мати на увазі, що у цьому випадку використаний зворотний порядок запису бітів двійкового числа, а не прямий, як раніше. Зверніть особливу увагу на те, що одним із елементів

мультиплікативного поля Галуа $GF(2^3)$ є число 0.

Таблиця 3.19 – Правила додавання для скінченого поля Галуа $GF(2^3)$

+	α^0	α^1	α^2	α^3	α^4	α^5	α^6
α^0	0	α^3	α^6	α^1	α^5	α^4	α^2
α^1	α^3	0	α^4	α^0	α^2	α^6	α^5
α^2	α^6	α^4	0	α^5	α^1	α^3	α^0
α^3	α^1	α^0	α^5	0	α^6	α^2	α^4
α^4	α^5	α^2	α^1	α^6	0	α^0	α^3
α^5	α^4	α^5	α^3	α^2	α^0	0	α^1
α^6	α^2	α^6	α^0	α^4	α^3	α^1	0

Таблиця 3.20 – Правила множення для скінченного поля Галуа $GF(2^3)$

x	α^0	α^1	α^2	α^3	α^4	α^5	α^6
α^0	α^0	α^1	α^2	α^3	α^4	α^5	α^6
α^1	α^1	α^2	α^3	α^4	α^5	α^6	α^0
α^2	α^2	α^3	α^4	α^5	α^6	α^0	α^1
α^3	α^3	α^4	α^5	α^6	α^0	α^1	α^2
α^4	α^4	α^5	α^6	α^0	α^1	α^2	α^3
α^5	α^5	α^6	α^0	α^1	α^2	α^3	α^4
α^6	α^6	α^0	α^1	α^2	α^3	α^4	α^5

Таблиця 3.21 – Степені примітивного елемента α у полі Галуа $GF(2^3)$

Степені α	Десяткове подання	Двійкове поліноміальне подання		
		x^0	x^1	x^2
0	0	0	0	0
α^0	1	1	0	0
α^1	2	0	1	0
α^2	4	0	0	1
α^3	3	1	1	0
α^4	6	0	1	1
α^5	7	1	1	1
α^6	5	1	0	1
α^7	0	0	0	0

Наведемо приклад кодування повідомлення із трьох символів, кожен із яких містить три біта [33].

Приклад 3.45. Побудувати у полі Галуа $GF(2^3)$ код Ріда – Соломона, який виправляє подвійну помилку, для повідомлення із трьох символів, заданих двійковим кодом через степені примітивного елемента α: $\alpha^1 = 010$; $\alpha^3 = 110$; $\alpha^5 = 111$.

Послідовність процесу кодування буде такою.

1. Формуємо поліном повідомлення, який, для прикладу, що розглядається, має вигляд:

$$\mathbf{m}(x) = \alpha^1 + \alpha^3 x + \alpha^5 x^2.$$

2. Шукаємо твірний поліном. Враховуючи співвідношення (3.199), можна записати:

$$\mathbf{g}(x) = (x - \alpha)(x - \alpha^2)\,(x - \alpha^3)(x - \alpha^4) = \alpha^3 + \alpha^1 x + \alpha^0 x^2 + \alpha^3 x^3 + x^4.$$

3. Після множення поліному $\mathbf{m}(x)$ на $x^{2\cdot t} = x^4$, маємо:

$$\mathbf{m}_{\mathbf{m}}(x) = x^4\,\mathbf{m}(x) = \alpha^1 x^4 + \alpha^3 x^{5} + \alpha^5 x^6.$$

4. Ділимо модифікований вхідний поліном $\mathbf{m}_{\mathbf{m}}(x)$ на твірний поліном $\mathbf{g}(x)$. Для прикладу, який розглядається, маємо такий результат:

$$\mathbf{p}(x) = \alpha^0 + \alpha^2 x + \alpha^4 x^2 + \alpha^6 x^3;\ \mathbf{U}(x) = \alpha^0 + \alpha^2 x + \alpha^4 x^2 + \alpha^6 x^3 + \alpha^1 x^4 + \alpha^3 x^5 + \alpha^5 x^6,$$

де $\mathbf{p}(x)$ – остача від ділення, $\mathbf{U}(x)$ – результат кодування.

Зверніть особливу увагу на те, що у прикладі, який був розглянутий, використаний не прямий порядок запису розрядів числа, який зазвичай притаманний цифровому та поліноміальному поданню кодових послідовностей, а зворотний, який більше притаманний векторам.

Приклад 3.46. Припустимо, що під час передавання інформації кодова послідовність Ріда – Соломона, отримана у прикладі 3.45, була спотворена, і два прийнятих символи, третій та четвертий, містять помилки. Будемо вважати, що третій, контрольний символ, містить однобітову помилку в третьому розряді, яку можна виразити як α^2, а четвертий символ, символ повідомлення – трибітову помилкою, яку можна виразити як α^5.

Для прикладу, який розглядається, вектор помилки можна записати наступним чином:

$$E(x) = 0 + 0x + 0x^2 + \alpha^2 x^3 + \alpha^5 x^4 + 0x^5 + 0x^6 =$$
$$= (0\ 0\ 0) + (0\ 0\ 0)x + (0\ 0\ 0)x^2 + (0\ 0\ 1)x^3 + (1\ 1\ 1)x^4 +$$

$$+(0\ 0\ 0)x^5+(0\ 0\ 0)x^6.$$

Тоді поліном спотвореного кодового слова, виражений через степені примітивного елемента α, має вигляд:

$$C(x)=(1\ 0\ 0)+(0\ 0\ 1)x+(0\ 1\ 1)x^2+(1\ 0\ 0)x^3+(1\ 0\ 1)x^4+(1\ 1\ 0)x^5+$$
$$+(1\ 1\ 1)x^6=\alpha^0+\alpha^2x+\alpha^4x^2+\alpha^0x^3+\alpha^6x^4+\alpha^3x^5+\alpha^5x^6.$$

За умови відомого поліному $C(x)$, розв'яжемо рівняння синдромів помилок через степені примітивного елемента α. Для формування такого розв'язку використаємо таблиці 3.19 та 3.20.

$$\begin{aligned}
S_1 &= C(\alpha)=\alpha^0+\alpha^3+\alpha^6+\alpha^3+\alpha^{10}+\alpha^8+\alpha^{11}= \\
&=\alpha^0+\alpha^3+\alpha^6+\alpha^3+\alpha^2+\alpha^1+\alpha^4=\alpha^3\neq 0, \\
S_2 &= C(\alpha^2)=\alpha^0+\alpha^4+\alpha^8+\alpha^6+\alpha^{14}+\alpha^{13}+\alpha^{17}= \\
&=\alpha^0+\alpha^4+\alpha^1+\alpha^6+\alpha^0+\alpha^6+\alpha^3=\alpha^5\neq 0, \\
S_3 &= C(\alpha^3)=\alpha^0+\alpha^5+\alpha^{10}+\alpha^9+\alpha^{18}+\alpha^{18}+\alpha^{23}= \\
&=\alpha^0+\alpha^5+\alpha^3+\alpha^2+\alpha^4+\alpha^4+\alpha^2=\alpha^6. \\
S_4 &= C(\alpha^4)=\alpha^0+\alpha^6+\alpha^{12}+\alpha^{12}+\alpha^{22}+\alpha^{23}+\alpha^{29}= \\
&=\alpha^0+\alpha^5+\alpha^3+\alpha^2+\alpha^4+\alpha^4+\alpha^2=\alpha^6\neq 0, \\
&=\alpha^0+\alpha^6+\alpha^5+\alpha^5+\alpha^1+\alpha^2+\alpha^1=0.
\end{aligned} \tag{3.248}$$

Зрозуміло, що після шостої степені значення степенів повторюються, починаючи з нуля, що цілком відповідає теорії циклотомічних класів [48]. Слід відзначити, що формула (3.210) дає аналогічний результат. Це зайвий раз підтверджує те, що теорія полів Галуа є цільною та єдиною, хоча в ній існують еквівалентні описи для багатьох алгебраїчних операцій.

Запишемо матричне рівняння для пошуку коефіцієнтів поліному локаторів помилок, задане співвідношеннями (3.220), (3.221) у вигляді:

$$\begin{pmatrix} S_1 & S_2 \\ S_2 & S_3 \end{pmatrix}\cdot\begin{pmatrix} \Lambda_1 \\ \Lambda_2 \end{pmatrix}=\begin{pmatrix} S_3 \\ S_4 \end{pmatrix},$$

або

$$\begin{pmatrix} \Lambda_1 \\ \Lambda_2 \end{pmatrix}=\begin{pmatrix} S_1 & S_2 \\ S_2 & S_3 \end{pmatrix}^{-1}\cdot\begin{pmatrix} S_3 \\ S_4 \end{pmatrix}. \tag{3.249}$$

Особливо слід відмітити, що у попередніх прикладах вектор

коефіцієнтів поліному локаторів помилок був записаний у вигляді $\begin{pmatrix}\Lambda_2\\ \Lambda_1\end{pmatrix}$.

Зміна порядку слідування елементів вектора **Λ** пояснюється тим, що у даному прикладі прямий порядок слідування розрядів був з самого початку замінений на зворотний.

Розв'яжемо отримане матричне рівняння (3.248) через обчислення визначників та алгебраїчних доповнень, тобто, з використанням співвідношень (3.244), (3.245).

Враховуючи значення синдромів помилок (3.248), відповідно маємо:

$$\begin{pmatrix}S_1 & S_2\\ S_2 & S_3\end{pmatrix}^{-1} = \frac{\begin{pmatrix}S_3 & S_2\\ S_2 & S_1\end{pmatrix}}{\det\begin{pmatrix}S_1 & S_2\\ S_2 & S_3\end{pmatrix}} = \frac{\begin{pmatrix}\alpha^6 & \alpha^5\\ \alpha^5 & \alpha^3\end{pmatrix}}{\alpha^5} = \alpha^2 \cdot \begin{pmatrix}\alpha^6 & \alpha^5\\ \alpha^5 & \alpha^3\end{pmatrix} =$$

$$= \begin{pmatrix}\alpha^8 & \alpha^7\\ \alpha^7 & \alpha^5\end{pmatrix} = \begin{pmatrix}\alpha^1 & \alpha^0\\ \alpha^0 & \alpha^5\end{pmatrix}. \qquad (3.250)$$

Розв'язок матричного рівняння (3.249), з урахуванням отриманого виразу для зворотної матриці (3.250) та значень синдромів помилок (3.248), має наступний вигляд:

$$\begin{pmatrix}\Lambda_1\\ \Lambda_2\end{pmatrix} = \begin{pmatrix}\alpha^1 & \alpha^0\\ \alpha^0 & \alpha^5\end{pmatrix} \cdot \begin{pmatrix}\alpha^6\\ 0\end{pmatrix} = \begin{pmatrix}\alpha^7\\ \alpha^6\end{pmatrix} = \begin{pmatrix}\alpha^0\\ \alpha^6\end{pmatrix},$$

тобто, $\Lambda_1 = \alpha^0$, $\Lambda_2 = \alpha^6$.

Тоді поліном локаторів помилок записується у вигляді:

$$\Lambda(x) = \alpha^0 + \Lambda_1 x + \Lambda_2 x^2 = \alpha^0 + \alpha^6 x + \alpha^0 x^2. \qquad (3.251)$$

Через підстановку коренів утворювального поліному $\mathbf{g}(x)$ до рівняння (3.251) отримуємо вказівники на ті позиції, на яких розташовані помилки. Вказівник на помилку фіксується лише у тому випадку, коли $\sigma(\alpha^i) = 0$. Виконаємо відповідні обчислення:

$\Lambda(\alpha^0) = \alpha^0 + \alpha^6 + \alpha^0 = \alpha^6 \neq 0$, помилка відсутня.

$\Lambda(\alpha^1) = \alpha^2 + \alpha^7 + \alpha^0 = \alpha^2 \neq 0$, помилка відсутня.

$\Lambda(\alpha^2) = \alpha^4 + \alpha^8 + \alpha^0 = \alpha^6 \neq 0$, помилка відсутня.

$\Lambda(\alpha^3) = \alpha^6 + \alpha^9 + \alpha^0 = 0$, помилка існує.

$\Lambda(\alpha^4) = \alpha^8 + \alpha^{10} + \alpha^0 = 0$, помилка існує.

$\Lambda(\alpha^5) = \alpha^{10} + \alpha^{11} + \alpha^0 = \alpha^2 \neq 0$, помилка відсутня.

$\Lambda(\alpha^6) = \alpha^{12} + \alpha^{12} + \alpha^0 = \alpha^0 \neq 0$, помилка відсутня.

Тоді значення номерів помилкових розрядів, згідно із таблицею 3.20, визначаються через примітивний елемент α наступним чином:

$$\alpha^4 = 1/u_1,\ u_1 = \alpha^3;\ \alpha^3 = 1/u_2,\ u_2 = \alpha^4, \quad j_1 = 3,\ j_2 = 4. \tag{3.252}$$

де j_1 та j_2 – значення номерів помилкових розрядів у поліномі $C(x)$, а u_1 та u_2 – синдроми помилок, які використовуються для подальших розрахунків.

Тепер врахуємо взаємозв'язок між синдромами помилок та вектором помилки, який за умови подвійної помилки можна записати у вигляді:

$$S_1 = C(\alpha) = e_1 u_1 + e_2 u_2; \qquad S_2 = C(\alpha^2) = e_1 u_1^2 + e_2 u_2^2, \tag{3.253}$$

де e_1, e_2 – компоненти вектора помилки, які відповідають значенням j_1 та j_2.

У матричній формі система рівнянь (3.253) записується наступним чином:

$$\begin{pmatrix} u_1 & u_2 \\ u_1^2 & u_2^2 \end{pmatrix} \cdot \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} S_1 \\ S_2 \end{pmatrix}, \tag{3.254}$$

або, через степені кореня твірного поліному α, враховуючи (3.252):

$$\begin{pmatrix} \alpha^3 & \alpha^4 \\ \alpha^6 & \alpha^8 \end{pmatrix} \cdot \begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} \alpha^3 \\ \alpha^5 \end{pmatrix}. \tag{3.255}$$

Матричне рівняння (3.254) є окремим випадком загального рівняння теорії кодування

$$\begin{pmatrix} u_1 & u_2 & \dots & u_t \\ u_1^2 & u_2^2 & \dots & u_t^2 \\ \dots & \dots & \dots & \dots \\ u_1^t & u_2^t & \dots & u_t^t \end{pmatrix} \cdot \begin{pmatrix} e_1 \\ e_2 \\ \dots \\ e_t \end{pmatrix} = \begin{pmatrix} S_1 \\ S_2 \\ \dots \\ S_t \end{pmatrix}, \tag{3.256}$$

де t – кількість помилок, які виправляються. Зрозуміло, що рівнянню (3.254) відповідає значення $t = 2$. Також не важко зрозуміти, що матричне рівняння (3.256) є аналогом рівняння (3.68), а матриця із значеннями $u_k^l, k \le t, l \le t$ відповідає матриці Вандермонда. Тобто, алгоритм декодування кодів Ріда – Соломона, який розглядається у цьому прикладі, є повним аналогом алгоритму

ПГЦ для кодів БЧХ, розглянутому у підрозділі 3.3.4.2.

Розв'язок матричного рівняння (3.255) отримуємо з використанням співвідношень (3.244), (3.245) у наступному вигляді:

$$\begin{pmatrix} \alpha^3 & \alpha^4 \\ \alpha^6 & \alpha^1 \end{pmatrix}^{-1} = \frac{\begin{pmatrix} \alpha^1 & \alpha^4 \\ \alpha^6 & \alpha^3 \end{pmatrix}}{\alpha^3\alpha^1 - \alpha^6\alpha^4} = \frac{\begin{pmatrix} \alpha^1 & \alpha^4 \\ \alpha^6 & \alpha^3 \end{pmatrix}}{\alpha^4 + \alpha^3} = \alpha^{-6}\begin{pmatrix} \alpha^1 & \alpha^4 \\ \alpha^6 & \alpha^3 \end{pmatrix} =$$

$$= \alpha^1\begin{pmatrix} \alpha^1 & \alpha^4 \\ \alpha^6 & \alpha^3 \end{pmatrix} = \begin{pmatrix} \alpha^2 & \alpha^5 \\ \alpha^7 & \alpha^4 \end{pmatrix} = \begin{pmatrix} \alpha^2 & \alpha^5 \\ \alpha^0 & \alpha^4 \end{pmatrix}.$$

Тоді розв'язок матричного рівняння (3.254), який дає нам значення елементів вектора помилки, буде наступним:

$$\begin{pmatrix} e_1 \\ e_2 \end{pmatrix} = \begin{pmatrix} \alpha^2 & \alpha^5 \\ \alpha^0 & \alpha^0 \end{pmatrix} \cdot \begin{pmatrix} \alpha^3 \\ \alpha^5 \end{pmatrix} = \begin{pmatrix} \alpha^5 + \alpha^{10} \\ \alpha^3 + \alpha^9 \end{pmatrix} = \begin{pmatrix} \alpha^2 \\ \alpha^5 \end{pmatrix}. \qquad (3.257)$$

Тобто, вектор помилки для прикладу, який розглядається, у поліноміальній формі має наступний вигляд:

$$E(x) = \alpha^2 x^3 + \alpha^5 x^4 = (0\ 0\ 0) + (0\ 0\ 0)x + (0\ 0\ 0)x^2 + (0\ 0\ 1)x^3 +$$

$$+ (1\ 1\ 1)x^4 + (0\ 0\ 0)x^5 + (0\ 0\ 0)x^6.$$

Сумування за модулем два вектора помилки $E(x)$ із спотвореною кодовою комбінацією $C(x)$ дає наступний результат:

$$F(x) = E(x) \oplus C(x) = ((0\ 0\ 0) \oplus (1\ 0\ 0)) + ((0\ 0\ 0) \oplus (0\ 0\ 1)) \cdot x +$$

$$+ ((0\ 0\ 0) \oplus (0\ 1\ 1)) \cdot x^2 + ((0\ 0\ 1) \oplus (1\ 0\ 0)) \cdot x^3 + ((1\ 1\ 1) \oplus (1\ 0\ 1)) \cdot x^4 +$$

$$+ ((0\ 0\ 0) \oplus (1\ 1\ 0)) \cdot x^5 + ((0\ 0\ 0) \oplus (1\ 1\ 1)) \cdot x^6 = (0\ 0\ 1) \cdot x + (0\ 1\ 1) \cdot x^2 +$$

$$+ (1\ 0\ 1) \cdot x^3 + (0\ 1\ 0) \cdot x^4 + (1\ 1\ 0) \cdot x^5 + (1\ 1\ 1) \cdot x^6.$$

Зрозуміло, що остаточний результат обчислень $F(x)$ відповідає поліному $\mathbf{U}(x)$, який був отриманий у прикладі 3.45.

Враховуючи те, що сформоване кодове слово призначено для виправлення подвійних помилок та відкидаючи перші чотири розряди із поліному $F(x)$, отримуємо закодоване інформаційне слово:

$$m(x) = [(0\ 1\ 0)\ (1\ 1\ 0)\ (1\ 1\ 1)].$$

Слід відзначити, що існує безпосередній зв'язок між двійковими та

багатопозиційними кодами Ріда – Соломона, відрізняється лише форма подання елементів поля Галуа. Наприклад, отримана послідовність $m(x)$ може бути переписана в еквівалентній формі $m(x)=[2, 3, 7]$. Для переведення двійкових чисел до десяткової системи може бути використана програма **conv2bin**, наведена у додатку Н, а для переведення десяткових чисел до двійкової системи – відповідно, програма **convbin2dec**.

Наведений приклад 3.46 зайвий раз підтверджує, що кількість помилок, які виправляють групові коди, визначається кількістю спотворених символів повідомлення, а не кількістю спотворених бітів. Єдина проблема полягає у тому, що, згідно із співвідношенням (3.234) та властивістю 3.1, довжина кодової послідовності *B* не повинна перевищувати значення максимального елементу поля $2^m - 1$. Наприклад, для коду, який розглядався у прикладах 3.45 та 3.46, визначена максимальна кількість помилок, що виявляються на виправляються, $t = 2$, для поля Галуа $GF(2^3)$ є найбільшою граничною величиною. Дійсно, код має 7 розрядів, а $2^m - 1 = 7$. У цьому випадку існує єдиний спосіб поліпшення коректувальних параметрів коду – це збільшення порядку поля Галуа. Але, як було відмічено у підрозділі 3.4.9, цей спосіб пов'язаний із суттєвим ускладненням завдання декодування та із значним зростанням часу проведення розрахунків. Узагальнені теоретичні оцінки коректувальних параметрів кодів Ріда – Соломона будуть наведені у підрозділі 3.12.

Крім цього, наведений приклад 3.46 зайвий раз підтверджує, що для вірного читання та обробки визначених кодових комбінацій вкрай важливим є врахування порядку слідування розрядів. Якщо, як у прикладі 3.46, задано не прямий, а зворотний порядок слідування розрядів, необхідно строго його притримуватись на протязі проведення всіх розрахунків.

3.4.11 Узагальнене описання розглянутих способів формування систематичних кодів Ріда – Соломона та декодування їхніх послідовностей

Розглянуті у попередніх підрозділах методи формування кодів Ріда – Соломона та декодування їхніх кодових послідовностей є досить складними та різноманітними. Тому у цьому підрозділі наведемо узагальнене

описання розглянутих вище алгоритмів, що у деякій мірі полегшує їх розуміння.

По-перш за все слід відзначити, що всі алгебраїчні операції, які виконуються для формування кодів Ріда – Соломона та декодування їхніх послідовностей, виконуються у полі Галуа $GF(2^m)$ з використанням співвідношень (3.210) – (3.218). Способи виконання елементарних алгебраїчних операцій у полях Галуа $GF(2^m)$ розглядалися у прикладах 3.27 – 3.31, а поліноміальних операцій – у прикладах 3.32 – 3.36.

Формування систематичних кодів Ріда – Соломона є досить простою процедурою, яка містить лише 4 кроки.

1. За умови визначеної кількості помилок t, які необхідно виявляти та виправляти, формується твірний поліном $g(x)$, заданий співвідношенням (3.200). Спосіб формування твірних поліномів у полях Галуа $GF(2^m)$ для різних значень параметра m та кількості помилок t був розглянутий у прикладі 3.37.

2. До інформаційного слова $M(x)$, яке необхідно закодувати, дописується зправа $2 \cdot t$ нулів. В результаті створюється модифіковане інформаційне слово $M_m(x)$. Ця алгебраїчна операція може бути записана як добуток полінома, який відповідає інформаційному слову $M(x)$, на поліном $x^{2 \cdot t}$, тобто $M_m(x) = M(x) \cdot x^{2 \cdot t}$. Також слід відзначити, що нулі дописуються зправа у разі прямого порядку запису розрядів числа $M(x)$, а у разі зворотного – зліва.

3. Знаходиться остача $R(x)$ від ділення поліному $M_m(x)$ на твірний поліном $g(x)$, тобто, $R(x) = \mathrm{mod}_{g(x)}(M_m(x))$.

4. Остаточна кодова послідовність $C(x)$ визначається як алгебраїчна сума поліномів $M_m(x) \oplus R(x)$ у полі Галуа $GF(2^m)$.

Способи формування послідовностей кодів Ріда – Соломона за описаним алгоритмом були розглянуті у прикладах 3.38 – 3.41.

Більш складним є розуміння різних способів декодування кодів Ріда – Соломона та пошуку помилкових розрядів. Оскільки існує декілька таких

способів і обрання того або іншого способу декодування залежить від кількості помилок, які виправляються кодом, розглянемо їх окремо та послідовно.

1. У будь-якому випадку необхідно через підстановку значень 2^i, $i = 1...2 \cdot t$, в поліном $C(x)$, знайти значення синдромів помилок S_i.

2. Якщо всі значення S_i дорівнюють 0, вважається, що у послідовності $C(x)$ помилка відсутня. У цьому випадку для отримання закодованого інформаційного слова із кодової послідовності $C(x)$ необхідно відкинути $2 \cdot t$ молодших розрядів.

3. Якщо будь-які із значень S_i не дорівнюють 0, необхідно через розв'язок матричного рівняння (3.220), (3.221) знайти коефіцієнти поліному локаторів помилок. Проте, оскільки це рівняння є досить складним, існує декілька способів його розв'язування, які були розглянуті вище. Перелічимо нижче головні з них.

3.1 Якщо розглядається кодова послідовність, яка дозволяє виправляти одну помилку, для пошуку номера помилкового розряду може бути використане співвідношення (3.237), а для пошуку значення помилки – співвідношення (3.241). Таким чином формується вектор помилки та знаходиться правильна кодова комбінація $F(x)$. Для отримання закодованого інформаційного слова із кодової послідовності $F(x)$ необхідно просто відкинути $2 \cdot t$ молодших розрядів. Спосіб пошуку одиночної помилки у систематичних кодах Ріда – Соломона був розглянутий у прикладі 3.43.

3.2 Якщо розглядається кодова послідовність, яка дозволяє виправляти декілька помилок, але значення кількості помилок t не перевищує 3, для розв'язування системи рівнянь (3.220), (3.221), можуть бути використані методи матричного аналізу, наприклад, пошук визначника та обчислення зворотної матриці або метод Гаусса – Зейделя. Узагальнений спосіб обчислення зворотної матриці визначається співвідношеннями (3.244), яке для матриці другого порядку записується у спрощеному вигляді (3.245). Способи проведення

відповідних розрахунків були розглянуті у прикладах 3.44 та 3.46.

3.3 Якщо розглядається кодова послідовність, яка дозволяє виправляти велику кількість помилок, більшу за 3, тоді матричні методи розрахунку стають неефективними. У цьому разі найбільш придатним для обчислення коефіцієнтів поліному локаторів помилок є алгоритм Берлекемпа – Мессі, блок схема якого наведена на рис. 3.8. Спосіб обчислення коефіцієнтів полінома локатору помилок з використанням алгоритму Берлекемпа – Мессі був розглянутий у прикладі 3.42. Також алгоритм Берлекемпа – Мессі реалізований у комп'ютерній програмі, наведеній у додатку О. Ця програма дозволяє виявляти та виправляти до 6 помилок у кодах Ріда – Соломона, побудованих у полях Галуа $GF(2^m)$ за умови $m = 3, 4, 5, 6, 7, 8, 12$ або 16.

4. Розраховуються значення поліному локаторів помилок від аргументу 2^{-u}, де $u = 1...n$ – номери розрядів кодової послідовності. Розряди, для яких $\Lambda(2^{-u}) = 0$, надалі вважаються помилковими.

5. Для пошуку значень елементів вектора помилок розв'язується матричне рівняння (3.254). За умови $t \le 3$ для розв'язування цього матричного рівняння можуть бути використані матричного аналізу, наприклад, пошук визначника, алгебраїчних доповнень та обчислення зворотної матриці з використанням співвідношень (3.244). За умови $t > 3$ більш ефективним є алгоритм, який базується на теоремі Форні та задається співвідношеннями (3.226) – (3.230). Спосіб пошуку значень елементів вектора помилок з використанням методів матричного аналізу був розглянутий у прикладі 3.45, а з використанням теореми Форні – у прикладі 3.41. Також алгоритм, оснований на теоремі Форні, реалізований у комп'ютерній програмі, наведеній у додатку О.

6. Через сумування за модулем два елементів векторів спотвореної кодової послідовності $C(x)$ та вектора помилки $E(x)$ формується правильна послідовність $F(x)$. Для отримання закодованого інформаційного слова із кодової послідовності $F(x)$ необхідно відкинути $2 \cdot t$ молодших розрядів.

7. Якщо значення поліному локаторів помилок для всіх значень $u = 1...n$ є ненульовими, вважається, що кодову комбінацію неможливо поновити. У

комп’ютерних програмах за такої умови видається повідомлення про помилку, яка не виправляється.

Головний висновок полягає у тому, що у разі незначної величини максимальної кількості помилок t, які виявляються та виправляються кодом Ріда – Соломона, а саме $t \le 3$, можуть бути використані матричні методи розв’язування систем рівнянь (3.220), (3.221) та (3.254), проте за умови $t > 3$ більш ефективним є використання алгоритму Берлекемпа – Мессі та теореми Форні. У випадку $t > 3$ важливим фактором є також універсальність алгоритму Берлекемпа – Мессі, яка полягає у тому, що він за будь-яких умов формує поліном найменшої степені. Тобто, якщо коректувальна здатність коду складає $t = 5$, але у кодовій послідовності виникла лише одна помилка, за $2 \cdot t = 10$ ітерацій буде сформовано поліном локаторів помилок першого порядку. У разі використання матричних алгоритмів система рівнянь (3.220), (3.221), для випадку $\tau < t$ є виродженою і необхідно знову проводити обчислення для матриць меншого порядку, які містять меншу кількість контрольних сум. Узагальнене порівняння ефективності алгоритму Берлекемпа – Мессі та матричних методів було наведено у таблиці 3.18. Із проведеного аналізу зрозуміло, що якщо для ручних розрахунків та для невеликих значень t більш зручними можуть бути матричні методи, для універсальних комп’ютерних програмних засобів значно більш ефективними є алгоритм Берлекемпа – Мессі та теорема Форні. Тому саме ці методи розв’язування завдання декодування послідовностей кодів Ріда – Соломона були реалізовані у комп’ютерній програмі, наведеній у додатку О.

3.4.12 Апаратні цифрові електронні пристрої для формування кодів Ріда – Соломона та декодування їхніх послідовностей

У підрозділі 3.4.8 була описана структура програми **RSC**, призначеної для формування кодів Ріда – Соломона та декодування їхніх послідовностей, а також була оцінена ефективність роботи цієї програми для різних значень m порядку

поля Галуа $GF(2^m)$ з точки зору витрат машинного часу. Проведені дослідження дозволили зробити важливий висновок про те, що ефективність роботи програмних засобів за умови збільшення порядку поля m суттєво знижується. У цьому і полягає суттєвий недолік програмної реалізації кодерів та декодерів для послідовностей кодів Ріда – Соломона [33, 56, 57]. Річ у тому, що зазвичай програмовані кодери та декодери не можуть проводити алгебраїчні операції у полі Галуа паралельно для всіх розрядів символів повідомлення. Проте слід відзначити, що цей недолік вже не притаманний сучасним багатоядерним та багатопроцесорним обчислювальним системам [21 – 23].

На відміну від цього, у разі апаратної реалізації кодерів Ріда – Соломона через регістри зсуву, елементи кодової послідовності можуть оброблятися як послідовно, так і паралельно. За умови послідовного об'єднання регістрів зсуву декодування одного символу із кількістю розрядів m здійснюється за m тактів, а за умови паралельного – лише за один такт [33, 56, 57]. Крім цього, у разі апаратної реалізації поліноміальних операцій у полі Галуа $GF(2^m)$ відпадає необхідність комбінаторного перебирання значень степеневої функції для здійснення множення та ділення елементів поля. Тобто, головною перевагою апаратних засобів для формування та декодування послідовностей кодів Ріда – Соломона над програмними є їхня більш висока швидкодія. Проте важливою перевагою програмних засобів над апаратними є більша гнучкість універсальність таких засобів. Наприклад, за умови використання програмованих кодувальних та декодувальних пристроїв можна легко змінювати як порядок поля m, так і максимальну кількість помилок t, які виявляються та виправляються кодом, або коректувальну здатність коду. Доречи, саме таким чином побудована програма, яка наведена у додатку О. Крім того, кодер і декодер можуть бути об'єднані в одній програмі, що значно підвищує її функціональність.

Розглянемо головні принципи побудови апаратних кодерів Ріда – Соломона. Після k зсувів, де k – довжина вхідного інформаційного слова, на виході кодера формується частка від ділення вхідної послідовності на заданий твірний поліном, а у самому останньому регістрі – відповідно, остача від цього ділення. За такої умови розряди частки є останніми розрядами коду. Для апаратної реалізації операції ділення у полі Галуа визначеного порядку m елементи вхідної послідовності $M(x)$ m_i одночасно множаться на коефіцієнти твірного поліному g_i та сумуються за модулем два із значеннями всіх регістрів. Тобто, ділення поліномів у полі Галуа здійснюється за схемою Горнера.

Аналогічний алгоритм ділення був розглянутий у прикладах 3.38 – 3.41 та реалізований у програмі, наведеній у додатку О, через функцію **`divpolgf2m`**, код якої наведений у додатку Н. Відповідна структурна схема кодера Ріда – Соломона наведена на рис. 3.45 [56, 57].

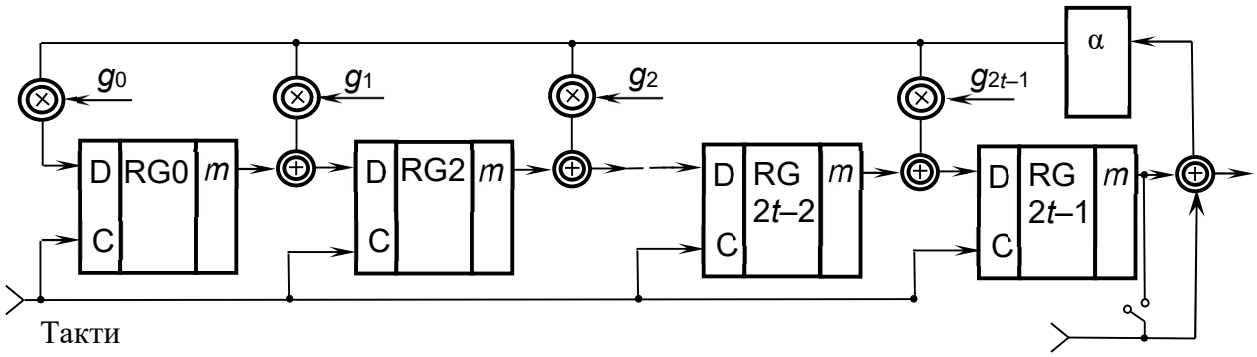


Рис. 3.45 Спрощена структурна схема кодера Ріда – Соломона.

Умовні позначення: ⊕ – сумування у полі Галуа $GF(2^m)$, ⊗ – множення у полі Галуа $GF(2^m)$

На рис. 3.46 наведена структурна схема кодера Ріда – Соломона, який створює код РС (255, 249) із виправленням потрійних помилок [56, 57]. Кожний із шести перевірочних розрядів містить символ довжиною 8 біт. Головними арифметичними операціями, які виконує кодер, є сумування за модулем два та множення на коефіцієнти твірного поліному у полі Галуа $GF(2^8)$.

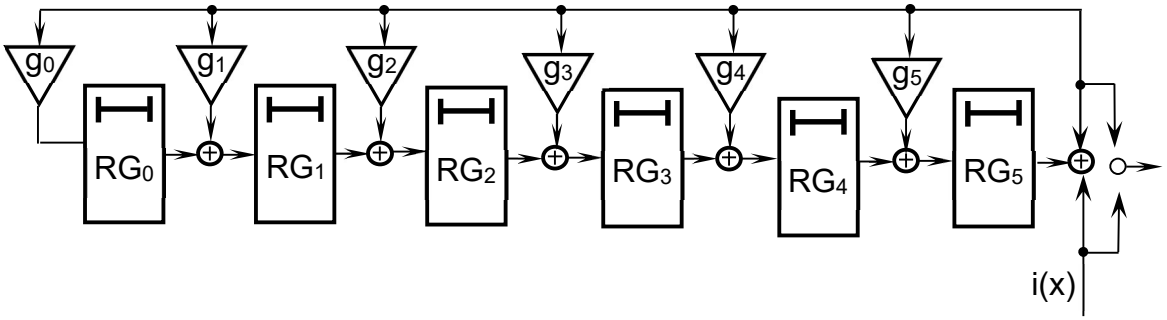


Рис. 3.46 Схема РС-кодера для формування коду (255, 249), який створюється у полі Галуа $GF(2^8)$

Щодо апаратної реалізації декодерів Ріда – Соломона – тут існує низка непостих технічних рішень [56, 57]. Загалом вони базуються на підтриманні різних мов програмування логічних мікросхем, насамперед це мови **VHDL** (абревіатура англійського словосполучення **VHSIC** (**V**ery **H**igh **S**peed

Integrated **C**ircuits) **H**ardware **D**escription **L**anguage) та Verilog [83]. Такий підхід є вельми ефективним та раціональним, оскільки гнучкість програмованих електронних систем дозволяє легко узгоджувати параметри декодера з іншими програмованими мікросхемами. Загалом цей підхід відповідає ідеології ПЛІС, яка була описана у шостому розділі першої частини посібника [1].

Щодо програмних реалізацій декодерів, тут головним недоліком є необхідність використання великої обчислювальної потужності для кодів із високою коректувальною здатністю. Крім цього, алгебраїчні операції у полях Галуа $GF(2^m)$ є нестандартними з обчислювальної точки зору, і зазвичай програмістам необхідно реалізовувати їх самостійно. Зокрема, таких функцій немає навіть у математичній бібліотеці системи науково-технічних розрахунків MatLab, яка, з початку XXI століття, вважається однією із самих потужних та повних [13, 14]. Тому саме такий підхід був використаний для написання комп'ютерної програми, наведеної у додатку О. Спочатку були створені обчислювальні процедури для реалізації елементарних та поліноміальних алгебраїчних операцій у полях Галуа $GF(2^m)$, наведені у додатку Н, а вже потім, на основі цих програмних модулей, реалізована програма для формування кодів Ріда – Соломона та декодування їхніх послідовностей. Наслідком складності виконання елементарних алгебраїчних операцій у полях Галуа є значний час проведення розрахунків, необхідний для декодування послідовностей кодів Ріда – Соломона, що досить довго у певній мірі стримувало розвиток програмних декодерів.

Проте, починаючи з дев'яностих років XX століття, за рахунок оптимального складання програм та зростання обчислювальних потужностей персональних комп'ютерів, з'явилась можливість створення ефективних програмних декодерів із високою швидкістю передавання інформації. Наприклад, для комп'ютерів з процесором Pentium з частотою роботи 166 МГц швидкість обробки та передавання інформації, закодованої РС-кодом (255, 251) становить 12 Мбіт/с, а у разі використання РС-коду (255, 233) – 1,1 Мбіт/с [56, 57]. Проте, у будь-якому разі, апаратні декодери є більш швидкодійними, особливо за умови паралельного підключення регістрів зсуву. Узагальнені схеми декодерів Ріда – Соломона наведені у підручнику [33] та у монографії [52], а низка практичних схем – у підручниках [56, 57]. Важливим параметром таких схем є параметр чергування j, який визначає кількість бітів, що можуть

перемежовуватися, змінюючи свою послідовність [56, 57]. Узагальнені схеми апаратних декодерів Ріда – Соломона для різних параметрів послідовностей, які декодуються, наведені на рис. 3.47 – 3.50.

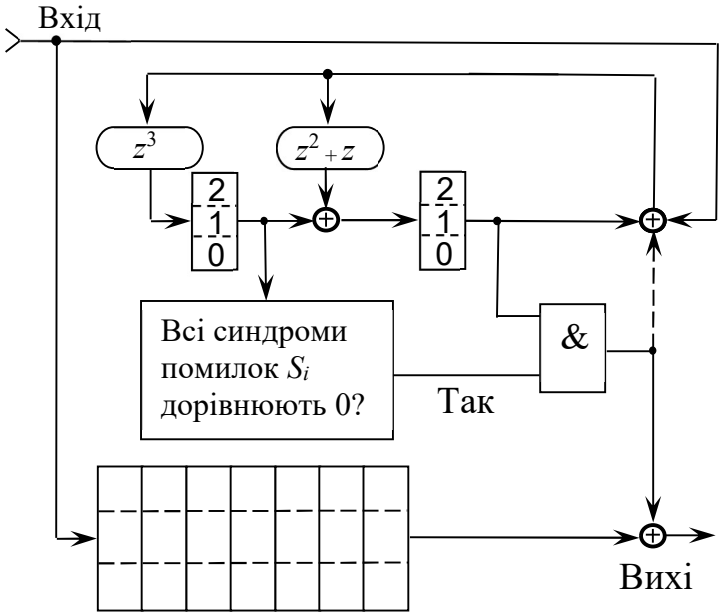


Рис. 3.47 Апаратна реалізація декодера коду Ріда – Соломона, який виправляє одиночну помилку, для поля Галуа $GF(2^3)$ [57]

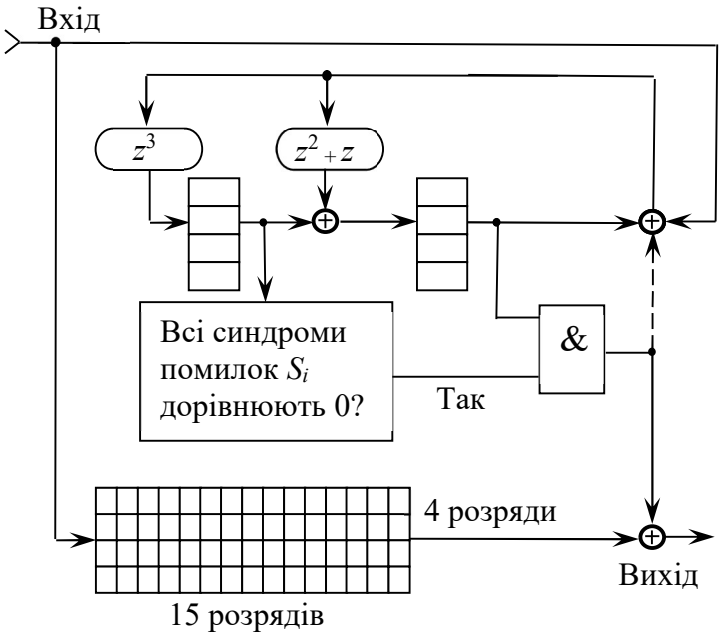


Рис. 3.48 Апаратна реалізація декодера коду Ріда – Соломона, який

виправляє подвійну помилку, для поля Галуа $GF(2^3)$ [57]

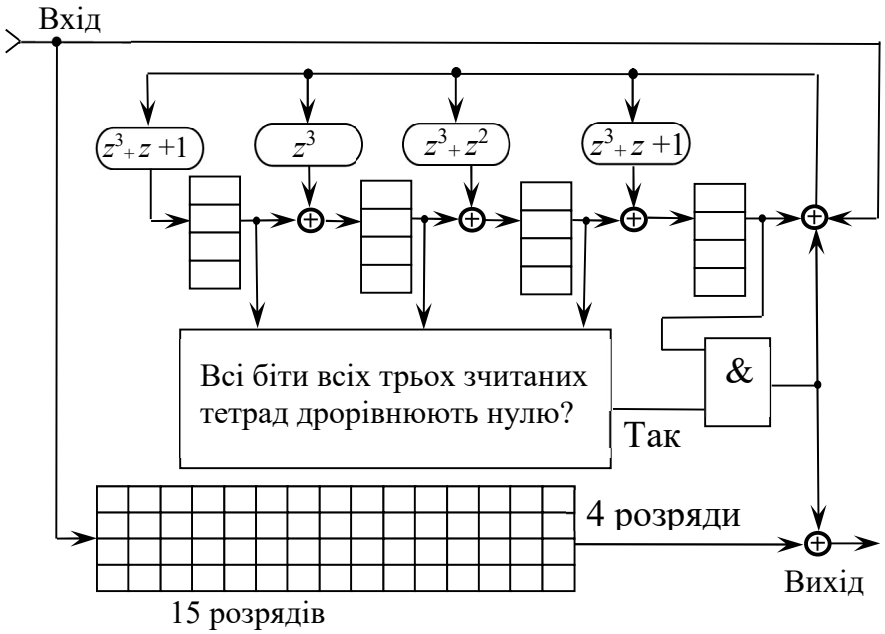


Рис. 3.49 Апаратна реалізація декодера коду Ріда – Соломона, який виправляє подвійну помилку, для поля Галуа $GF(2^4)$ [57]

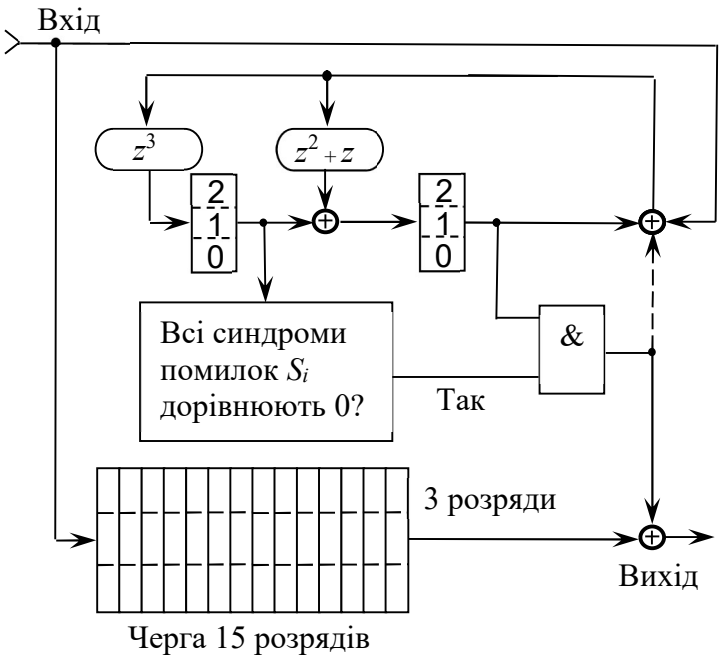


Рис. 3.50 Апаратна реалізація декодера коду Ріда – Соломона, який виправляє

подвійну помилку, для поля Галуа $GF(2^3)$ за умови значення параметра чергування $j = 2$ [57]

3.4.13 Оцінка залежності коректувальної здатності кодів Ріда – Соломона від довжини кодової послідовності та від порядку поля Галуа

Перед вивченням цього підрозділу необхідно повторити підрозділи 1.4, 6.1.3 та 6.1.4 другої частини посібника

У підрозділі 3.4.2 були наведені співвідношення (3.206), (3.207), для оцінки ймовірності хибного приймання кодової комбінації коду Ріда – Соломона. Проте ці співвідношення є лише частковим випадком загальної ймовірнісної оцінки коректувальних параметрів кодів Ріда – Соломона, оскільки вони є правильними лише за умови $n = 2^m - 1$, де n – довжина кодової комбінації, m – порядок поля Галуа.

Скористаємося узагальненими оцінками правильного та хибного приймання цифрових повідомлень, які були наведені у підрозділі 6.1.4 другої частини посібника [49]. Згідно із законами теорії ймовірностей будемо вважати, що імовірність спотворення одного символу повідомлення за умови відомої ймовірності бітової помилки p_b складає [1, 8, 49]:

$$p_m = 1 - (1 - p_b)^m, \tag{3.258}$$

де m – порядок поля Галуа та, відповідно, кількість бітів у одному символі.

Тоді, згідно із законом біноміального розподілу Бернуллі [49], можна записати імовірність того, що у повідомлені із n символів будуть спотворені рівно θ бітів [48, 49]:

$$P(\tau = \theta) = C_n^{\theta} \cdot \left(1 - (1 - p_b)^m\right)^{\theta} \cdot \left((1 - p_b)^m\right)^{n-\theta}; \qquad C_n^{\theta} = \frac{n!}{\theta! \cdot (n - \theta)!}. \tag{3.259}$$

де C_n^{θ} – кількість сполучень із n елементів по θ [48].

Враховуючи отримане співвідношення (3.259), а також те, що код Ріда – Соломона може виправляти t помилок, використовуючи закон сумування ймовірностей незалежних подій [49], можна обчислити імовірність правильного та хибного приймання послідовності з n символів. Для імовірності правильного приймання повідомлення відповідно маємо [81]:

$$P_{\text{П}} = P(\tau \le \theta) = \sum_{\theta=0}^{t} C_n^{\theta} \cdot \left(1 - (1 - p_b)^m\right)^{\theta} \cdot \left((1 - p_b)^m\right)^{n-\theta}, \; t \le n, \tag{3.260}$$

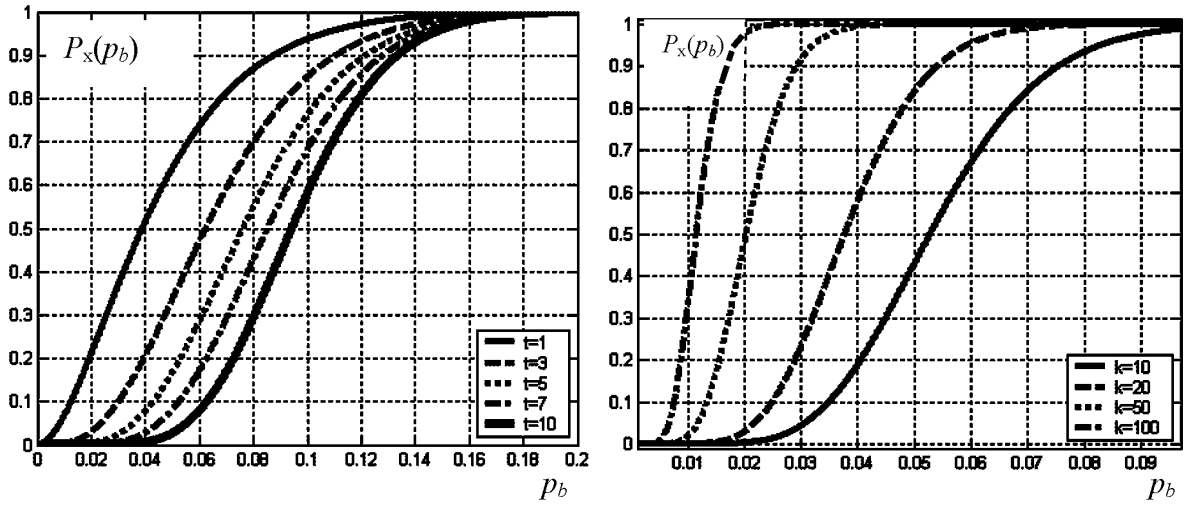
а для імовірності хибного приймання повідомлення [81]:

$$P_{\text{х}} = 1 - P_{\text{п}} = P(\tau > \theta) = \sum_{\theta=t+1}^{n} C_n^{\theta} \cdot \left(1-(1-p_b)^m\right)^{\theta} \cdot \left((1-p_b)^m\right)^{n-\theta}, t < n. \quad (3.261)$$

Зрозуміло, що за випадок $t = n$ є не реальним. Також зрозуміло, що для кодів Ріда – Соломона $n = k + 2 \cdot t$, де k – кількість інформаційних символів у повідомленні, і що, як завжди, необхідною умовою існування коду є виконання граничного співвідношення (3.234). Тоді максимальне значення t визначається через довжину кодової комбінації n наступним чином:

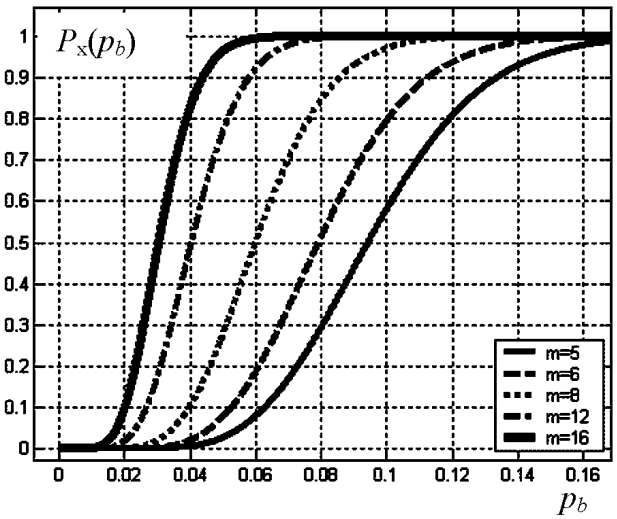
$$t_{\max} = \begin{cases} \dfrac{n-1}{2}, & \text{за умови непарних значень } n; \\ \dfrac{n}{2} - 1, & \text{за умови парних значень } n. \end{cases} \quad (3.262)$$

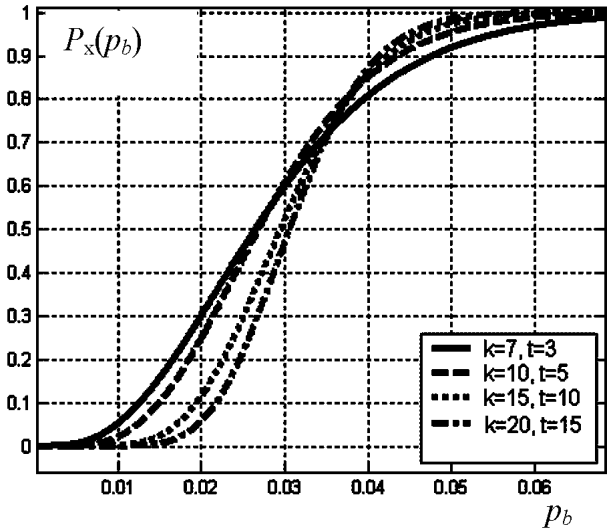
Залежності імовірності хибного приймання кодової комбінації $P_{\text{х}}$ від імовірності бітової помилки p_b та від параметрів коду, отримані через співвідношення (3.261) з використанням засобів програмування системи MatLab, наведені на рис. 3.51, а – г.



а)

б)





в) г)

Рис 3.51 Залежності імовірності помилки у коді Ріда – Соломона, яку не можливо виправити, від параметрів коду: а) $m = 5$, $k = 7$; б) $m = 8$, $t = 10$; в) $k = 7$, $t = 10$; г) $m = 12$

Із графічних залежностей, наведений на рис. 3.51, можна зробити наступні висновки. У будь-якому разі залежності $P_{\text{х}}(p_b)$ є зростаючими функціями, які за умови $p_b \to 1$ асимптотично наближаються до одиниці. Цей факт можна легко пояснити тим, що за умови зростання імовірності бітової помилки збільшується і ймовірність хибного приймання загальної кодової комбінації, але цілком зрозуміло, що за законами теорії ймовірностей значення $P_{\text{х}}$ не може перевищувати 1 [49]. Щодо залежностей функцій $P_{\text{х}}(p_b)$ від параметрів кодових комбінацій m, k та t, вони також мають певні закономірності, які можна обґрунтувати. Із збільшенням значень параметрів m та k за умови постійного значення t імовірність хибного приймання кодової комбінації $P_{\text{х}}$ зростає, що обумовлено зростанням загальної довжини кодової комбінації у бітах N_b, яка обчислюється як добуток порядку поля m та довжину кодової комбінації у символах n, тобто:

$$N_b = m \cdot n = m \cdot (k + 2 \cdot t). \qquad (3.263)$$

Оскільки за визначеною умовою максимальна кількість помилкових символів t, які виявляються та виправляються, є постійною величиною, імовірність хибного приймання кодової комбінації $P_{\text{х}}$ закономірно збільшується. Збільшення величини $P_{\text{х}}$ із зростанням кількості інформаційних символів повідомлення k є цілком зрозумілим та не потребує особливих пояснень, відповідні графічні залежності наведені на рис. 3.51, б. Зростання $P_{\text{х}}$ із збільшенням порядку поля m для залежностей, наведених на рис. 3.51, в, обумовлено збільшенням довжини кодової комбінації за загальною кількістю бітів, проте цей результат моделювання є більш цікавим та потребує окремого глибокого аналізу, який буде проведений нижче.

Із залежностей, наведених на рис. 3.51, а, видно, що величина $P_{\text{х}}$ із зростанням максимальної кількості помилок у символах коду t, які виявляються та виправляються, не збільшується, а навпаки, зменшується, не зважаючи на те, що довжина кодової послідовності у бітах N_b, задана співвідношенням (3.263), за умови зростання t також збільшується. Така закономірність пов'язана з тим, що, хоча у разі зростання величини t на 1 величина N_b збільшується на $2 \cdot m$, проте також збільшується на 1 кількість символів повідомлення, які можуть бути виправлені. За такої умови сума у рівнянні (3.261) має меншу кількість складових, тому значення $P_{\text{х}}$ із зростанням параметра t стабільно зменшується.

Якщо графічні залежності, наведені на рис. 3.51, а – в, досить легко пояснити та обґрунтувати з теоретичної точки зору, то залежності, наведені на рис. 3.51, г, вже є не настільки зрозумілими, і тому вони більш цікаві для поглибленого аналізу. Оскільки для залежностей, наведених на рис. 3.51, г, одночасно зростають і параметр k, і параметр t, поведінка функцій $P_{\text{х}}(p_b)$ за такої умови не є настільки передбаченою, і графіки, наведені на рис. 3.51, г, перетинаються. У разі малих значень імовірності бітової помилки p_b найбільшим значенням $P_{\text{х}}$ відповідають більш короткі кодові комбінації, що однозначно пов'язано із низькими значеннями параметра t. Кількість бітових помилок у таких системах зв'язку є не дуже високою, і тому за умови великих значень t майже всі помилки у переданих символах виправляються, а імовірність хибного приймання кодових комбінацій суттєво зменшується, майже до нуля.

Проте за умови більш високих значень імовірності бітової помилки p_b, які відповідають діапазону $p_b > 0{,}035$, максимальним значенням $P_{\text{х}}$ відповідають довгі кодові послідовності із високими значеннями параметра t. Це можна пояснити тим, що за умови потужних завад у каналі зв'язку кількість бітових помилок у символах переданого повідомлення стає вкрай високою, і,

навіть у випадку великих значень t, ці помилки кодом не виправляються.

Тобто , у разі великих значень p_b, зазвичай не кількість помилок t, які виправляються кодом, а саме довжина кодової комбінації у бітах N_b стає більш важливим фактором, який впливає на імовірнісні коректувальні параметри кодів Ріда – Соломона. У будь-якому разі, імовірнісні оцінки параметрів кодів Ріда – Соломона за формулами (3.260), (3.261), з урахуванням імовірності бітової помилки у каналі зв'язку p_b, є вкрай важливими та повинні братися до уваги інженерами під час проектування складних кодувальних електронних систем.

Тому ще раз повернемося до графіків, наведених на рис. 3.51, в, які описують залежність імовірності хибного приймання кодової послідовності від порядку поля Галуа. Якщо дійсно із збільшенням порядку поля зростає імовірність приймання спотвореної кодової комбінації, як це показують залежності, наведені на рис. 3.51, в, тоді виникає доречне запитання. А може, насправді, варто не збільшувати, а зменшувати порядок полів Галуа для побудови послідовностей кодів Ріда – Соломона? Цей висновок підтверджується також тим фактом, що, згідно із аналізом часу формування послідовностей кодів Ріда – Соломона та їхнього декодування, який був проведений у підрозділі 3.4.8, збільшення порядку поля Галуа дуже суттєво знижує швидкість проведення обчислень. Чому ж, у такому разі, у сучасній кодувальній та обчислювальній електронній апаратурі намагаються переходити до полів Галуа більш високих порядків [33, 56, 57]?

Щоб розібратися у поставленому не простому питанні, повернемося до співвідношення (3.234), яке визначає граничну величину для максимальної кількості помилок t, які виявляються та виправляються кодом. Розглянемо простий випадок, m = 3. Поле Галуа $GF(2^3)$ задано на множині із восьми символів, від 0 до 7. У такому полі можна сформувати три типа кодів, а саме:

– код (7,5), який виправляє одиночні помилки, t = 1. Кількість

інформаційних розрядів k у такому коді не повинна перевищувати значення 5, тобто, $k \le 5$;

– код (7,3), який виправляє подвійні помилки, t = 2. Кількість інформаційних розрядів у такому коді не повинна перевищувати значення 3, тобто, $k \le 3$;

– код (7,1), який виправляє потрійні помилки, t = 3. У такому коді може бути лише 1 інформаційний розряд, тобто, $k = 1$.

Слід відзначити, що спосіб формування коду Ріда – Соломона (7,3) у полі $GF(2^3)$ та алгоритм пошуку подвійної помилки у такому коді були розглянуті у прикладах 3.45 та 3.46, наведених у підрозділі 3.4.9.

Тобто, у полі Галуа низького порядку $GF(2^3)$ можна формувати лише коди Ріда – Соломона із виправленням одиночних та подвійних помилок. Код із виправленням потрійної помилки у полі Галуа $GF(2^3)$ містить лише 1 інформаційний символ, тобто, він є вкрай надлишковим, і такий код можна розглядати лише суто теоретично. Безперечним є те, що у разі формування коду Ріда – Соломона у полі Галуа низького порядку $GF(2^3)$ має виконуватись співвідношення $t \le 3$, яке є наслідком узагальненого співвідношення (3.234).

Тепер підвищимо порядок поля до значення m = 4, тобто, будемо формувати код у полі Галуа $GF(2^4)$. Оскільки поле Галуа $GF(2^4)$ задано на множині не із восьми, а із шістнадцяти символів, від 0 до 16, кількість можливих способів формування коду Ріда – Соломона помітно збільшується. У цьому випадку значення максимальної кількості помилок t, які виявляються та виправляються кодом, може складати $t = \left[\frac{2^4 - 1}{2}\right] = 7$, а це у 2,5 разів більше, ніж для поля $GF(2^3)$. Слід відзначити, що за умови t = 7 код Ріда – Соломона у полі Галуа $GF(2^4)$ містить лише 1 інформаційний символ і також є вкрай надлишковим для практичного застосування. Тому перелічимо лише типи кодів із меншою кількістю контрольних символів та визначимо їх коректувальні параметри.

1. (15, 13), $t = 1$; 2. (15, 11), $t = 2$; 3. (15, 9), $t = 3$; 4. (15, 7), $t = 4$;

5. (15, 5), $t = 5$; 6. (15, 3), $t = 6$.

Тобто, з використанням кодів Ріда – Соломона над полем Галуа $GF(2^4)$ з'являється реальна можливість виправляти не дві, а шість помилок. Зрозуміло, що за такої умови зростає довжина кодової комбінації, як у символах, так і у бітах. Проте, як показуються залежності, наведені на рис. 3.51, а, у разі зростання величини t імовірність приймання хибного повідомлення не збільшується, а зменшується. А що стосується ускладнення завдань кодування та декодування з обчислювальної точки зору, то, як видно із результатів, наведених у додатку О, у разі зміни параметру m із значення 3 на значення 4 час проведення обчислень сильно не змінюється. Тестові розрахунки показали, що суттєве зростання часу проведення обчислень $T_{\text{об}}$ спостерігається лише за умови $m \geq 8$.

У загальному випадку, для поля Галуа $GF(2^m)$, згідно із співвідношеннями (3.234) та (3.263), максимальне значення t становить:

$$t_{\max_m} = \left[\frac{2^m - 1}{2}\right] = 2^{m-1} - 1. \qquad (3.264)$$

Співвідношення (3.264) відповідає значенню $k = 1$. За умови $k > 1$ можна записати:

$$t_{\max_{m,k}} = \left[\frac{2^m - k - 1}{2}\right] = \left[2^{m-1} - \frac{k+1}{2}\right] = 2^{m-1} - \left[\frac{k+1}{2}\right] =$$

$$= \begin{cases} 2^{m-1} - \dfrac{k}{2} - 1, & \text{за умови парних значень } k; \\ 2^{m-1} - \dfrac{k+1}{2}, & \text{за умови непарних значень } k. \end{cases} \qquad (3.265)$$

Залежності імовірності хибного приймання кодової комбінації коректувального коду Ріда – Соломона від порядку поля m та від кількості помилок t, які виявляються та виправляються, наведені на рис. 3.52.

Проаналізуємо графічні залежності, які наведені на рис. 3.52. Початково був взятий код із параметрами $k = 21$ та $t = 5$ у полі Галуа $GF(2^5)$. Зрозуміло,

що для такого поля параметр t, згідно із формулою (3.265), має максимальне значення. За такої умови існує лише один шлях поліпшення коректувальної здатності коду – це підвищення порядку поля Галуа, тому друга залежність отримана також для коду із параметрами $k = 21$ та $t = 5$, але у полі Галуа $GF(2^6)$. Як і для залежностей, наведених на рис. 3.51, в, за такої умови коректувальна здатність коду не поліпшується, а погіршується, оскільки кількість помилок, які виправляються, не змінилася, а довжина кодової комбінації у бітах N_b в результаті зміни порядку поля зросла на 31 біт. Проте перевага такого перетворення полягає у тому, що через незначне збільшення порядку поля Галуа з'являється можливість адресувати більшу кількість розрядів коду, вже не 31, а 63.

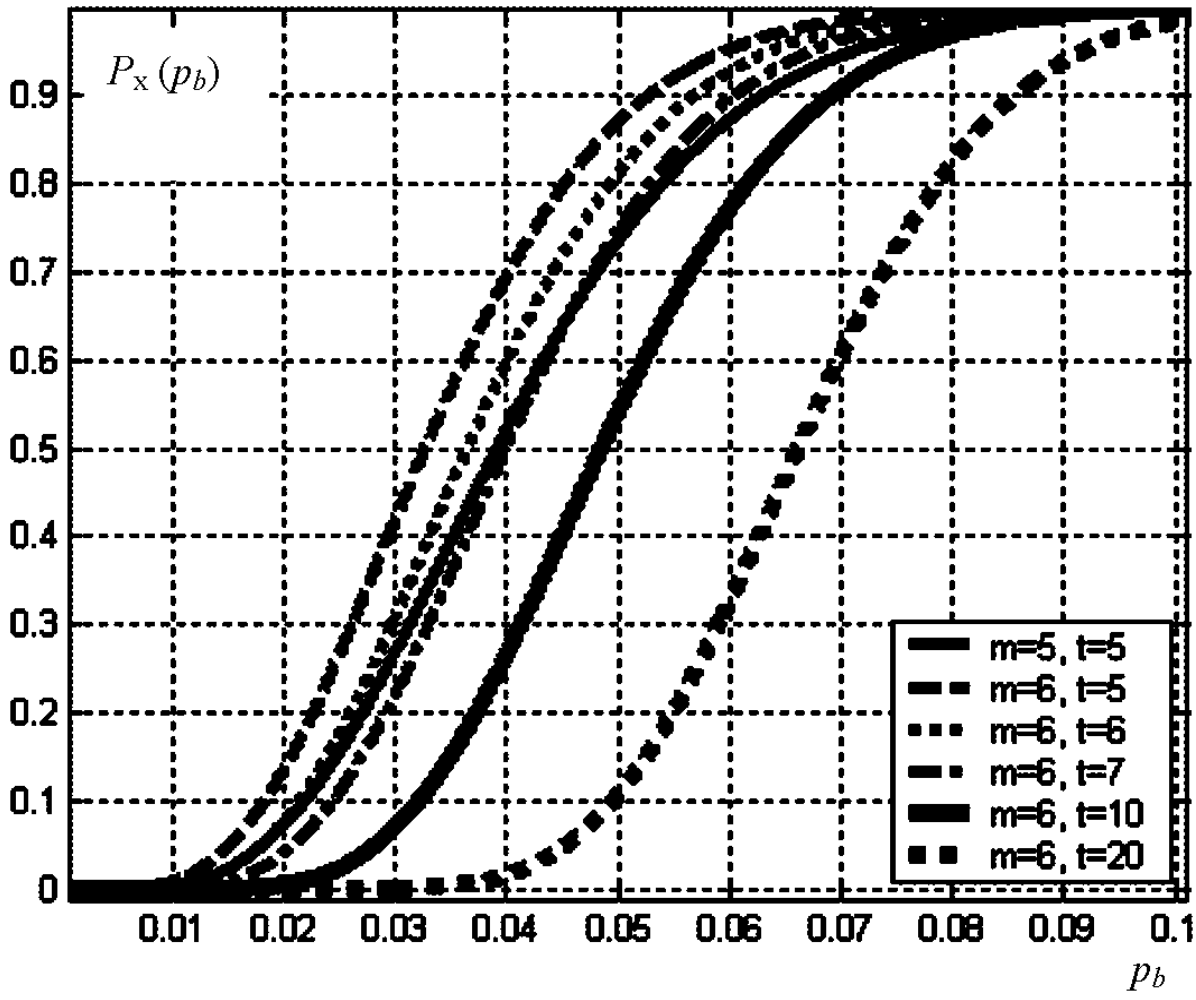


Рис. 3.52 Залежність імовірності появи у помілдовності коду Ріда – Соломона помилки, яку не можливо виправити, від порядку поля Галуа m та від

максимальної кількості помилок *t*, які виправляються, за умови кількості інформаційних символів $k = 21$

Тобто, тепер без зайвих проблем можна збільшувати значення *t*. Як показують залежності, наведені на рис. 3.52, за умови $t = 6$ коректувальна здатність коду (33, 21) у полі Галуа $GF(2^6)$ є дещо кращою, ніж для коду (31, 21) у тому ж полі, але гіршою, ніж для коду (31, 21) у полі Галуа $GF(2^5)$, який був розглянутий початково. Проте вже для коду (35, 21) у полі Галуа $GF(2^6)$, якому відповідає значення $t = 7$, імовірність хибного приймання повідомлень майже співпадає із відповідними значеннями для коду (31, 21) у полі Галуа $GF(2^5)$. Але у полі Галуа $GF(2^6)$ є можливим і подальше поліпшення коректувальної здатності коду. Наприклад, згідно із графічними залежностями, наведеними на рис. 3.52, для значення бітової помилки $p_b = 0{,}05$ за умови $t = 7$ маємо досить високу величину імовірності спотворення кодової комбінації $P_{\text{х}} \approx 0{,}75$, значенню $t = 10$ відповідає величина $P_{\text{х}} \approx 0{,}55$, а за умови $t = 20$ отримуємо досить низьке значення імовірності хибного приймання повідомлення $P_{\text{х}} \approx 0{,}1$.

Код програми **RSERROR**, яка була використана для побудови графічних залежностей, наведених на рис. 3.51, 3.52, наведений у додатку О.

Також слід відзначити, що коректувальна здатність кодів Ріда – Соломона є дещо вищою, ніж у кодів БЧХ, розглянутих у підрозділі 3.3. Наприклад, код БЧХ (15, 7) виправляє подвійну бітову помилку, а відповідний код Ріда – Соломона у полі Галуа $GF(2^4)$ – до чотирьох помилок у символах. Код БЧХ (15, 5) виправляє потрійну бітову помилку, а відповідний код Ріда – Соломона у полі Галуа $GF(2^4)$ – до п’яти помилок у символах.

Для розрахунків параметрів надлишковості кодів Ріда – Соломона можуть бути використані максимальні оцінки для кодів БЧХ, задані співвідношеннями (3.24), що свідчить про оптимальність кодових послідовностей Ріда – Соломона з точки зору теорії завадостійкого кодування.

Слід відзначити, що для кодів Ріда – Соломона із високим значенням

коефіцієнта надлишковості зростає також час передавання інформації в системах зв'язку. Проте, з іншого боку, з урахуванням того, що завадостійкий код виправляє помилки, у багатьох випадках відпадає необхідність у зайвому повторному передаванні інформації. Інформаційні оцінки параметрів систем зв'язку із надлишковими кодами є предметом окремих досліджень, пов'язаних із теорією інформації та кодування [1 – 10, 84]. Зрозуміло, що кількість додаткової інформації, яку необхідно передавати, насамперед пов'язана із надлишковістю коду та із імовірністю спотворення кодових комбінацій, яка визначається співвідношенням (3.261). Приклади відповідних інформаційних оцінок були наведені у навчальному посібнику [1] та у підрозділі 2.1 цієї частини посібника.

Також із графічних залежностей, які наведені на рис. 3.51, 3.52, можна зробити висновок, що коди Ріда – Соломона є вкрай ефективними за умови малих значень імовірності бітової помилки p_b, проте у разі високих значень цього параметру, близьких до 1, імовірність хибного приймання кодової комбінації значно підвищується і груповий код Ріда – Соломона втрачає свою ефективність. За умови високого значення імовірності бітової помилки p_b через наявність потужних завад у каналі зв'язку більш ефективним стає використання згорткових кодів, принципи побудови яких будуть описані у підрозділі 3.8.

Тобто, для аналізу ефективності використання групових кодів необхідно враховувати інформаційіні параметри каналів зв'язку, імовірність появи шумів та, відповідно, імовірність виникнення бітових помилок. Насправді, саме через імовірність виникнення бітової помилки p_b проводиться аналіз ефективності використання групового коду через обчислення імовірності хибного приймання всієї кодової комбінації $P_{\text{х}}$ з використанням співвідношення (3.261). За умови високих значень $P_{\text{х}}$ можливо зменшення цієї величини через збільшення максимальної кількості помилок t, які виправляються кодом, у полях Галуа більш високого порядку m. Відповідні теоретичні відомості щодо оцінки інофрмаційних параметрів каналів зв'язку були розглянуті у першому, другому, четвертому та п'ятому підрозділах першої частини посібника [1], а математичний апарат теорії груп – у другому та третьому розділах другої частини посібника [48]. Способи формування та

використання імовірносних оцінок у теорії кодування сигналів розглядалися у шостому розділі другої частини посібника [49].

3.5 Каскадні коди

У підрозділах 3.3 та 3.4, як приклади групових кодів, були розглянуті коди БЧХ та Ріда – Соломона. Також у цих підрозділах були описані алгоритми та відповідні апаратні та програмні цифрові електронні засоби, призначені для формування цих типів кодів та декодування їхніх послідовностей. Були наведені приклади формування цих типів групових кодів, як ручним способом, так і з використанням відповідних програмних засобів, наведених у додатках М та О. Із матеріалу, наведеного у підрозділах 3.3 та 3.4, зрозуміло, що загалом алгоритми формування групових кодів ґрунтуються на теорії поліномів та теорії груп, відповідний математичний апарат ґрунтується на методах дискретної математики та розглядався у другій частині посібника [48 – 50].

У підрозділі 3.4.12 була описана узагальнена методика оцінки ефективності кодів Ріда – Соломона на основі методів теорії імовірностей [49] та показано, що у разі обрання оптимальних параметрів цього коду його ефективність може бути досить високою. Тому, як було відмічено у підрозділі 3.1, сьогодні коди Ріда – Соломона часто використовуються у сучасній цифровій електронній апаратурі та у системах зв'язку, знаходять вони впровадження і в комп'ютерних системах.

Проте, з іншого боку, для проектування сучасних ефективних цифрових електронних систем, у яких кількість можливих помилок у повідомленнях, що приймаються, має бути вкрай обмеженою, використання описаних методів одноразового кодування інформації може не давати бажаного результату. Тому в сучасній цифровій електронній апаратурі все частіше використовуються методи подвійного кодування інформації, які дозволяють досягти більшої ефективності та надійності роботи цифрової кодувальної апаратури [63].

З теоретичної точки зору загальний принцип роботи системи із подвійним кодуванням оснований на тому, що спочатку інформаційне повідомлення кодується одним типом кодом, а потім отримане кодове слово

кодується кодом іншої структури, який має більшу коректувальну здатність. Зазвичай таке подвійне кодування ефективно використовується для побудови групових кодів великої довжини, оскільки, як було показано у підрозділі 3.4.12, для кодів Ріда – Соломона довжина кодової послідовності завжди є обмеженою розмірністю поля Галуа [63].

Розглянемо узагальнений спосіб подвійного кодування інформаційного повідомлення. Припустимо, що на вхід кодувальної системи надходить інформаційне повідомлення І1. Це повідомлення кодується кодом К1, в результаті чого формується кодове слово КС1. Після цього кодом К2, який розрахований на більшу довжину кодової комбінації, кодується кодове слово КС1, в результаті чого отримується інше кодове слово КС2, яке має більшу довжину, ніж КС1. Саме слово КС2 і передається через канал зв'язку. Зрозуміло, що на виході каналу зв'язку необхідно здійснити зворотний процес декодування. Спочатку кодове слово КС2 необхідно декодувати з використанням декодера коду К2 ДК2, в результаті чого має бути отримане кодове слово КС1. Після цього кодове слово КС1 розшифровується з використанням декодера коду К1 ДК1. В результаті виконання цих двох операцій декодування має бути отримане початкове інформаційне повідомлення І1. Якщо на першому або другому етапі декодування виникає помилка, яка може бути виправленою, вона автоматично виправляється, а якщо помилка виявлена, але її не можна вправити – формується запит на повторне надсилання відповідного повідомлення І1. Згідно із теоретичними відомостями, наведеними у підрозділі 3.1, такий підхід є загальним для формування та декодування блокових кодів.

Узагальнена структурна схема системи зв'язку із подвійним кодуванням наведена на рис. 3.53.

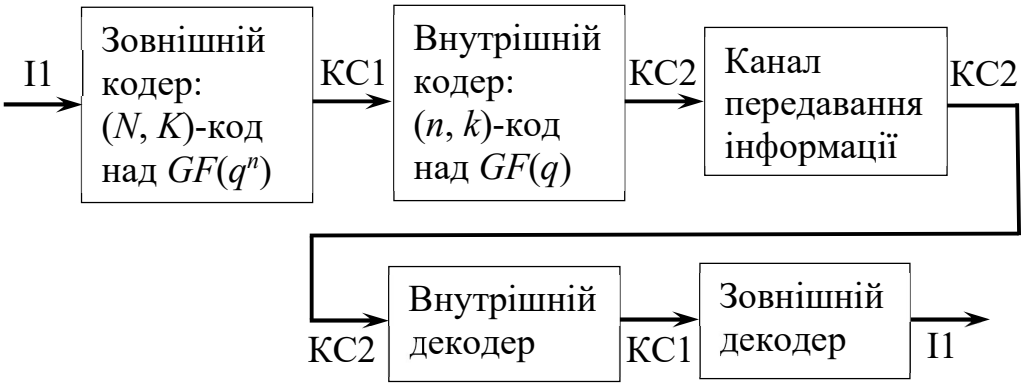


Рис. 3.53 Узагальнена структурна схема цифрового пристрою, призначеного для формування каскадного коду

Блокові коди, у яких використовується описаний вище спосіб подвійного кодування, у теорії кодувальних систем називаються каскадними, або гніздовими. Надамо відповідне визначення [63].

Визначення 3.13. Каскадним кодом (англійський термін – concatenated code, каскадний код, або nested code, гніздовий код), називається блоковий код, побудований як результат послідовного використання двох групових кодів із різною коректувальною здатністю.

Як відмічено у наведеному визначені, в англомовній літературі каскадний код називають по-різному, або каскадним, або гніздовим, проте у вітчизняній літературі загальновживаним є термін «каскадний код» [5].

Параметри блокових кодів обчислюються наступним чином.

1. Кількість інформаційних розрядів:

$$k_{\text{бл.}} = k_{\text{вн.}} \cdot k_{\text{зов.}} \tag{3.266}$$

2. Загальна кількість розрядів:

$$n_{\text{бл.}} = n_{\text{вн.}} \cdot n_{\text{зов.}} \tag{3.267}$$

В формулах (3.266), (3.267), індекси бл. відповідають блоковому коду,

індекси вн. – внутрішньому коду, а індекси зов. – зовнішньому.

Алгоритм побудови блокового коду має наступний вигляд [63].

1. Для вхідного інформаційного слова I1 будується зовнішній код із параметрами $m_1 = 2^q$, $k = k_1$, $t_1 = 2 \cdot k_1$. З використанням цього коду формується кодова послідовність КС1.

2. Для кодової послідовності КС1 формується внутрішній код із параметрами: $m_2 = 2^w$, $k = k_2$, $t_2 = 2 \cdot k_2$. Необхідною умовою формування внутрішнього коду є виконання співвідношень:

$$q > w, k_1 > k_2. \tag{3.268}$$

Слід відзначити, що загальна кількість помилок t, які виправляються, залежить від структури каскадного коду. Вона завжди є більшою, ніж значення t_1 та t_2, проте значно меншою за величину $\frac{r_{бл.}}{2}$, де $r_{бл.} = n_{бл.} - k_{бл.}$. Тобто, у загальному випадку для параметрів блокового коду завжди виконуються наступні тотожності:

$$r_{бл.} = n_{бл.} - k_{бл.}, t_{бл.} < r_{бл.} / 2. \tag{3.269}$$

У формулах (3.269) параметри $n_{бл.}$ та $k_{бл.}$ обчислюються через співвідношення (3.266) та (3.267).

Тобто, незважаючи на достатньо високу ефективність каскадних кодів з точки зору їхньої коректувальної здатності, головним їхнім недоліком є вкрай висока надлишковість. Така висока надлишковість каскадного коду пов’язана з тим, що внутрішнім кодом кодується не все кодове слово КС1, початково закодоване зовнішнім кодом, а його окремі символи.

Наочно структуру каскадного коду можна показати у вигляді квадратної матриці, наведеної на рис. 3.54. Зрозуміло, що кодове слово внутрішнього коду

формується єдиним вектором із довжиною $n_{вн.}$. Довжина символів на другому етапі кодування складає $k_{зов.}$. Тобто, завжди має місце тотожність:

$$L_{вн.} = k_{зов.}, \tag{3.270}$$

де $L_{вн.}$ – довжина кодового слова зовнішнього коду.

Слід також відзначити, що кількість інформаційних символів внутрішнього коду $k_{вн.}$ зазвичай не відповідає загальній кількісті символів першого коду $n_{зов.}$, оскільки для здійснення кодування на першому та другому етапах використовуються поля Галуа різних порядків.

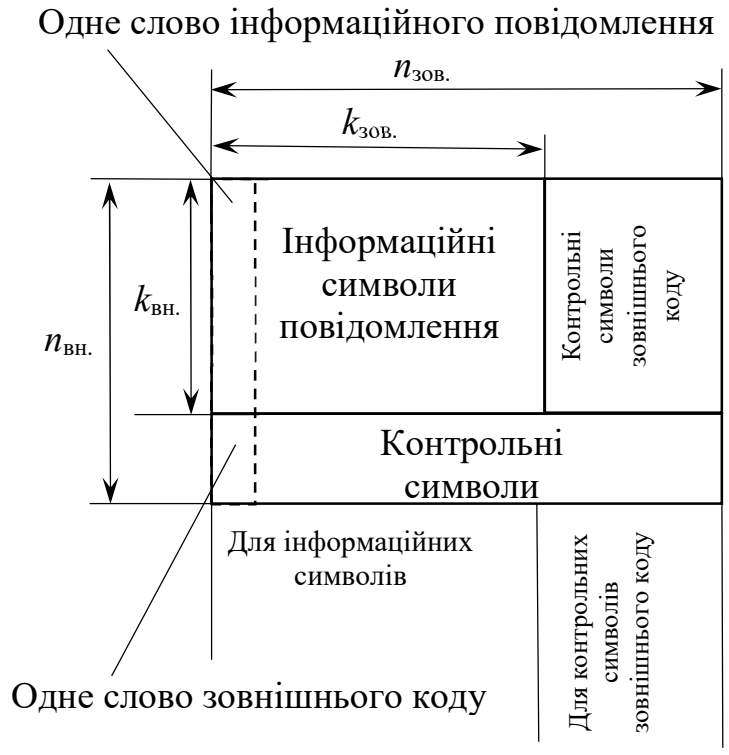


Рис. 3.54 Узагальнена структура каскадного коду та взаємозв’язок між його параметрами

Зрозуміло, що каскадні коди є узагальненим варіантом прямокутних кодів, розглянутих у підрозділі 2.3.3. Іноді в науковій літературі каскадні коди називають також ітеративними [5].

Зважаючи на це, коректувальну здатність каскадного коду можна

оцінити як:

$$t_{\text{бл.}} = \left[\frac{\sqrt{n_{\text{бл.}} - k_{\text{бл.}}}}{4}\right]. \quad (3.271)$$

Розглянемо приклад формування каскадного коду [63].

Приклад 3.47. Знайти параметри каскадного коду, якщо як внутрішній код використовується код РС(7, 3) над полем Галуа $GF(2^3)$, а як зовнішній – код РС(511, 505) над полем Галуа $GF(2^9)$.

Із загальної теорії кодів Ріда – Соломона, описаної у підрозділі 3.4.1, зрозуміло, що код РС(7, 3) виправляє подвійну помилку, а код РС(511, 505) – потрійну помилку.

Згідно із співвідношеннями (3.266), (3.267), результатом формування такого коду буде РС-код над полем Галуа $GF(2^3)$ із наступними параметрами:

1. $k_{\text{бл.}} = k_{\text{вн.}} \cdot k_{\text{зов}} = 3 \cdot 505 = 1515$.

2. $n_{\text{бл.}} = n_{\text{вн.}} \cdot n_{\text{зов.}} = 7 \cdot 511 = 3577$.

Тобто, параметри розглянутого каскадного коду Ріда – Соломона становлять РС(3577, 1515). Проте, незважаючи на досить велику кількість перевірочних символів, отримана коректувальна здатність каскадного коду РС(3577, 1515) не є вкрай високою. Він гарантовано виправляє 11 помилок, хоча для багатьох кодових послідовностей визначає й більшу їхню кількість. Значення $t_{\text{бл.}} = 11$ можна отримати із співвідношення (3.271).

Інші оцінки коректувальної здатності каскадного коду наведені у підручнику [5]. У разі формування набору контрольних символів через матрицю, за рядками та стовпчиками, мінімальна кількість помилок, які неможливо виявити, складає 4. Загальна кількість чотириразових помилок у багатопозиційному коді із розмірністю l x n складає [5]:

$$B_{4\text{з.}} = C_{nl}^4 = \frac{nl(nl-1)(nl-2)(nl-\ )}{4!}. \quad (3.272)$$

З іншого боку, для чотириразових кількості помилок, які виправляються двома контрольними символами, що стоять у стовпчику та у рядку, можна записати наступний математичний вираз [5]:

$$B_{4\text{в.}} = C_n^2 C_l^2 \frac{nl(n-1)(l-1)}{2!2!}. \quad (3.273)$$

Таким чином, співвідношення кількості чотириразових помилок, які виправляються каскадним кодом, до їхньої загальної кількості, згідно із

формулами (3.272), (3.273), визначається наступним математичним виразом [5]:

$$\frac{B_{4в.}}{B_{43.}} = \frac{6(n-1)(l-1)}{(nl-1)(nl-2)(nl-3)}. \quad (3.274)$$

Розглянемо інший приклад формування каскадного коду, в якому проілюструємо головні особливості його роботи [63].

Приклад 3.48. Побудувати каскадний код для наступної кодової послідовності із 72-х символів:

0 1 2 3 4 5 6 7 6 5 4 3 2 1 0 0 1 2
3 4 5 6 7 6 5 4 3 2 1 0 0 1 2 3 4 5
6 7 6 5 4 3 2 1 0 0 1 2 3 4 5 6 7 6
5 4 3 2 1 0 0 1 2 3 4 5 6 7 6 5 4 3

на основі зовнішнього коду РС(22, 18) над полем Галуа $GF(2^{12})$ та внутрішнього коду РС(7, 4) над полем Галуа $GF(2^3)$. Після цього занести помилки у сформований код, деякі символи вважати стертими, а потім перевірити, чи виправляє каскадний код помилкові та стерті символи.

Будемо розв'язувати завдання цього прикладу послідовно.

1. На основі символів вхідного повідомлення сформуємо кодове слово зовнішнього коду РС(22, 18) над полем Галуа $GF(2^{12})$.

0 4 6 5 | 0 1 2 3 4 5 6 7 6 5 4 3 2 1 0 0 1 2
2 4 6 5 | 3 4 5 6 7 6 5 4 3 2 1 0 0 1 2 3 4 5
4 0 0 2 | 6 7 6 5 4 3 2 1 0 0 1 2 3 4 5 6 7 6
2 1 7 2 | 5 4 3 2 1 0 0 1 2 3 4 5 6 7 6 5 4 3

Контрольні символи коду РС(22, 18) для чотирьох послідовностей із 18 символів обчислюються у поліноміальній формі згідно із співвідношенням (3.199).

2. Аналогічно формуємо кодові слова внутрішнього коду. Це буде 22 кодових слова коду РС(7, 4) над полем Галуа $GF(2^3)$. Записувати їх будемо у вигляді матриці, відповідно до рис. 3.54. Контрольні символи розташуємо зверху. Відповідна матриця буде мати наступний вигляд.

7 7 6 7 | 4 1 4 5 4 1 0 4 2 5 4 3 6 2 6 4 1 4
4 4 5 6 | 0 4 7 2 6 5 2 4 2 1 5 4 3 4 6 0 4 7
2 2 5 2 | 4 6 1 7 1 1 2 5 3 0 7 3 3 2 2 4 6 1
– – – – | – – – – – – – – – – – – – – – – – –

0 4 6 5 | 0 1 2 3 4 5 6 7 6 5 4 3 2 1 0 0 1 2

2 4 6 5 | 3 4 5 6 7 6 5 4 3 2 1 0 0 1 2 3 4 5

4 0 0 2 | 6 7 6 5 4 3 2 1 0 0 1 2 3 4 5 6 7 6

2 1 7 2 | 5 4 3 2 1 0 0 1 2 3 4 5 6 7 6 5 4 3

Структура побудованого каскадного коду показана на рис. 3.55. Зрозуміло, що загалом вона відповідає узагальненій структурі, яка відображена на рис. 3.54.

3. Тепер розглянемо особливості роботи сформованого каскадного коду. Спочатку занесемо в нього помилки. Припустимо, що прийнята кодова послідовність має наступний вигляд, де спотворені символи підкреслені, а замість затертих символів стоїть знак –. Оскільки тепер знак « – » використовується для позначення непрочитаного символу, рядок із цих знаків, який був використаний під час описання структури каскадного коду для відділення контрольних символів внутрішнього коду від інформаційних, тепер пропущений.

4 7 – 7 | 4 – 4 5 4 1 – 4 2 1 4 – 6 2 6 4 – 4

4 3 – – | 0 2 7 2 – – 2 4 2 4 5 4 3 4 5 0 4 7

2 2 – 2 | 4 4 1 7 1 1 2 5 3 5 7 3 3 – 2 4 6 1

0 4 – – | 1 6 0 – – – – – – – – – – – – – – – –

– – – – | – – – – – – – – – – – 1 0 0 1 2 2 4 5

4 0 – 2 | 6 1 6 5 4 3 2 1 – – – 4 – 0 5 6 – 6

2 1 5 2 | 5 2 3 – 1 0 0 1 2 6 4 5 6 7 6 5 4 1

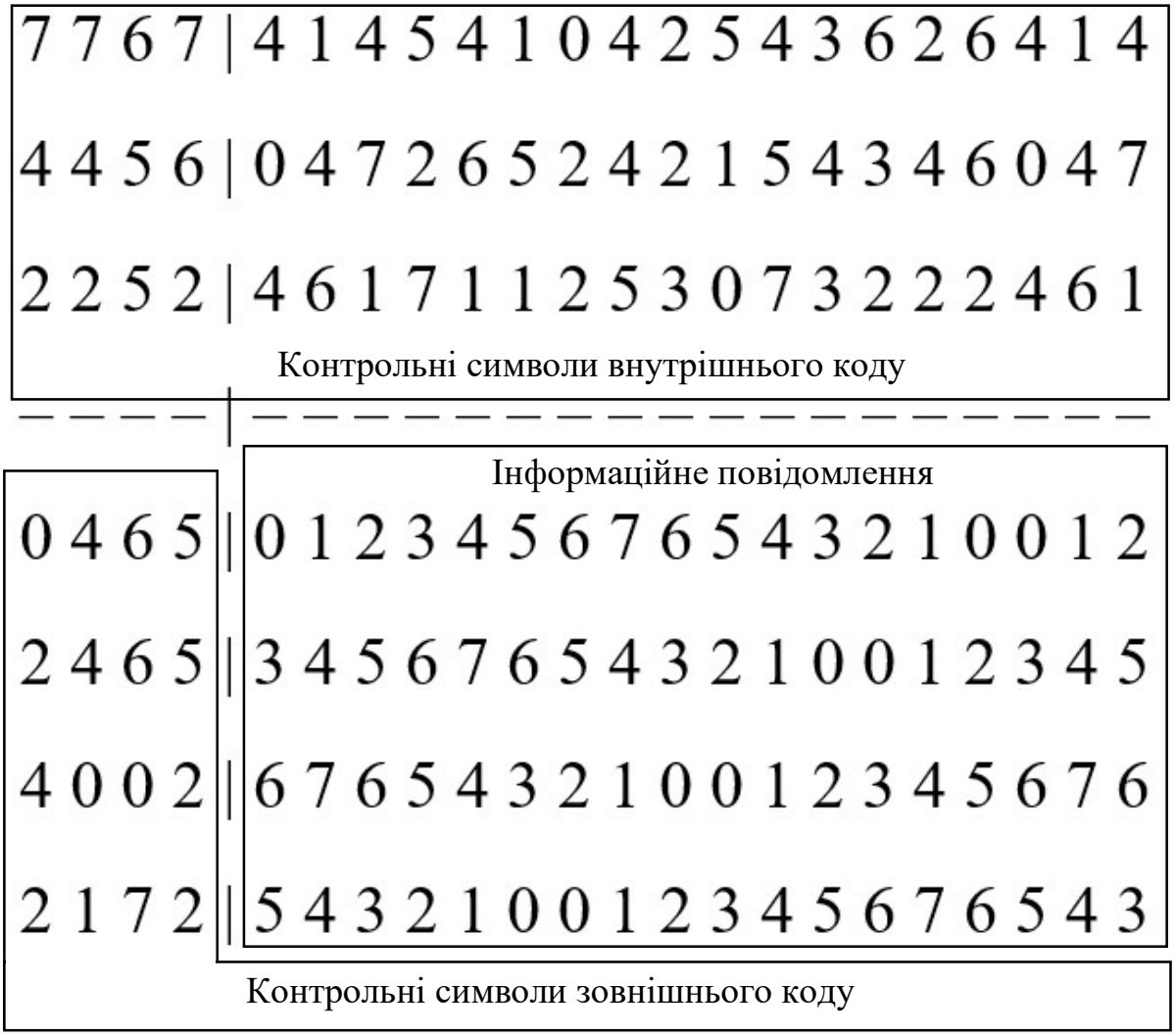


Рис. 3.55 Структура каскадного коду для прикладу 3.48

4. Тепер проаналізуємо, як спрацьовує каскадний код, виправляючи внесені помилки. Спочатку розглянемо роботу внутрішнього декодера. Зрозуміло, що перші три рядка, в яких містяться контрольні символи внутрішнього коду, після проведення операції обчислення контрольних сум за алгоритмом Берлекемпа – Мессі, описаним у підрозділі 3.4.6, можна ігнорувати. Оновлена числова послідовність буде мати наступний вигляд.

0 4 – 5 | 0 – 2 3 4 5 6 7 6 7 4 3 2 1 0 0 1 2

2 4 – 5 | 3 – 5 6 7 6 5 4 3 3 1 0 0 1 2 3 4 5

4 0 – 2 | 6 – 6 5 4 3 2 1 0 0 1 2 3 4 5 6 7 6

2 1 – 2 | 5 – 3 2 1 0 0 1 2 6 4 5 6 7 6 5 4 3

Тобто, після внутрішнього декодування більша частина помилкових та

невизначених символів коду вже поновлено.

5. Тепер розглянемо роботу внутрішнього декодера. Після декодування сформованих чотирьох рядків та відкидання контрольних символів оновлена числова послідовність буде мати наступний вигляд.

0 1 2 3 4 5 6 7 6 5 4 3 2 1 0 0 1 2
3 4 5 6 7 6 5 4 3 2 1 0 0 1 2 3 4 5
6 7 6 5 4 3 2 1 0 0 1 2 3 4 5 6 7 6
5 4 3 2 1 0 0 1 2 3 4 5 6 7 6 5 4 3

Тобто, вона повністю співпадає із початковою числовою послідовністю, яка була закодована каскадним кодом.

Із наведеного прикладу видно, що використання подвійної перевірки, за рядками та за стовпчиками, у значній мірі підвищує коректувальну здатність каскадного коду. Для здійснення внутрішнього кодування використовується канал із пам'яттю. Основи роботи систем зв'язку з каналом із пам'яттю та відповідні структурні схеми таких каналів зв'язку розглядалися у навчальних посібниках [1, 8]. Символи, які надходять з зовнішнього кодека, записуються до пам'яті каналу до тих пір, доки не буде сформоване кодове слово для внутрішнього кодека. Такий тип пам'яті у системах зв'язку називається пам'яттю із перемежуванням (англійський термін – corner-turning memory) [63].

Як було відмічено, незважаючи на досить високу коректувальну здатність, каскадні коди, сформовані на основі принципу кодування кодів Ріда – Соломона, є вкрай надлишковими. Тому на практиці частіше для формування каскадних кодів використовують комбінацію з кодів Ріда – Соломона та згорткових кодів. Основи теорії та практики використання згорткових кодів розглядатимуться у підрозділі 3.8. Існують також оптимальні каскадні коди, основані на алгоритмах формування кодів Ріда – Соломона. У таких кодах для зменшення довжини кодових послідовностей використовують інформаційні оцінки, основані на обчисленні ентропії повідомлення. Одним із прикладів такого коду є код Юстенсена [63].

Слід також відзначити, що каскадні коди дуже схожі на прямокутний код, який розглядався у підрозділі 2.3.3. Взагалі матричне подання кодових послідовностей є вкрай ефективним і часто використовується у практиці формування завадостійких кодів. На матричних алгоритмах оснований також спосіб формування іткративних кодів, які орзглядатимуться у підрозділі 3.7.1.

У наступному підрозділі розглянемо інші принципи формування групових кодів, які ґрунтуються на методах спектрального аналізу дискретних числових послідовностей. Відповідні розділи дискретної математики, зокрема дискретне перетворення Фур'є, розглядалися у першій та у другій частинах цього навчального посібника [1, 48, 50].

3.6 Спектральні методи формування групових кодів

3.6.1 Визначення дискретного перетворення Фур'є у полі Галуа та його властивості

Розглянемо вектор комплексних чисел $\mathbf{p} = \{p_i, i = 1, 2, \ldots, N-1\}$. Будемо вважати, що дійсні та умовні частини комплексних чисел, які є компонентами заданого вектору **p**, визначені у полі Галуа порядку q. Тоді для такого вектору можна застосовувати відому формулу дискретного перетворення Фур'є [1], відповідне аналітичне співвідношення можна записати наступним чином:

$$P_k = \sum_{i=0}^{N-1} \exp\left(-\frac{2\pi jik}{N}\right) p_i, \tag{3.275}$$

де $j = \sqrt{-1}$ – уявна одиниця.

У наведеному співвідношенні (3.275) для ядра дискретного перетворення Фур'є у полі Галуа порядку q має місце тотожність:

$$\exp\left(-\frac{2\pi jik}{N}\right) = \sqrt[N]{1}. \tag{3.276}$$

Введемо поняття характеристики поля [63].

Визначення 3.14. Характеристикою поля n поля Галуа порядку q називається кількість елементів його найменшого підполя.

Головну властивість дискретного перетворення Фур'є над комплексними числами, заданими як елементи поля Галуа $GF(q)$, можна сформулювати наступним чином [63].

Властивість 3.3. Якщо α є примітивним елементом поля Галуа $GF(q)$, а n – характеристичне число цього поля, тоді пряме дискретне перетворення Фур'є для елементів вектору **v** записується у вигляді співвідношення:

$$V_j = \sum_{i=0}^{n-1} \alpha^{ij} v_i. \tag{3.277}$$

Відповідно, співвідношення для зворотного дискретного перетворення Фур'є записується наступним чином:

$$v_j = \frac{1}{n} \sum_{i=0}^{n-1} \alpha^{-ij} V_i. \tag{3.278}$$

Розглянемо ще деякі важливі властивості дискретного перетворення

Фур'є у полі Галуа $GF(q)$ [63].

Властивість 3.4. (Властивість про згортку.) Якщо всі елементи вектору **e** є результатом множення елементів векторів **f** та **g**, тобто, має місце тотожність $e_i = f_i\, g_i$, тоді для елементів векторів **E**, **F** та **G**, які є результатом прямого перетворення Фур'є для векторів **e**, **f** та **g** відповідно, має місце тотожність:

$$E_j = \frac{1}{n}\sum_{k=0}^{n-1} F_{(j-k)} G_k, \tag{3.279}$$

де запис $(j-k)$ означає, що обчислення проводяться за арифметикою поля Галуа $GF(q)$.

Властивість 3.5. (Властивість зсуву.) Якщо вектори **v** та **V** є парою перетворення Фур'є, тобто $\{v_i\} \leftrightarrow \{V_j\}$, тоді парами перетворення Фур'є є також вектори

$$\{\alpha^i v_i\} \leftrightarrow \{V_{j+1}\}, \{v_{i-1}\} \leftrightarrow \{\alpha^j V_j\}. \tag{3.280}$$

Властивість 3.6. У загальному випадку результат дискретного перетворення Фур'є над полем Галуа $GF(q)$ можна записати у поліноміальному вигляді. Якщо вектор дискретних значень **v** заданий поліномом

$$v(x) = v_{n-1}x^{n-1} + v_{n-2}x^{n-2} + \ldots + v_1 x + v_0, \tag{3.281}$$

тоді результатом прямого дискретного перетворення Фур'є над вектором **v** є вектор

$$V(x) = V_{n-1}x^{n-1} + V_{n-2}x^{n-2} + \ldots + V_1 x + V_0. \tag{3.282}$$

Визначення 3.15. Вектор $V(x)$, заданий співвідношенням (3.282), називається спектральним, або асоційованим поліномом до поліному $v(x)$.

Важливими є дві наступні властивості записаних вище поліномів (3.281) та (3.282) [63].

Властивість 3.7. Елемент α^j є коренем поліному $v(x)$ тоді і лише тоді, коли $V_j = 0$.

Властивість 3.8. Елемент α^{-i} є коренем поліному $V(x)$ тоді і лише тоді, коли $v_j = 0$.

Тобто, згідно із властивостями 3.7 та 3.8, корінь поліному (3.280) відповідає нулю спектральної функції (3.277), а корінь спектрального

поліному (3.281) – нулю початкової послідовності v_j.

3.6.2 Обмеження спряженості для елементів дискретного спектру

Згідно із властивістю 3.3 та співвідношеннями (3.277), (3.278), дискретне перетворення Фур'є над полем Галуа $GF(q)$ приймає набір значень з іншого, розширеного поля $GF(q^m)$. Враховуючи це, зворотне перетворення Фур'є (3.278) не є ізоморфнім для набора значень початкового поля $GF(q)$. Формально це означає, що в результаті зворотного дискретного перетворення Фур'є можуть бути отримані числові значення із полів більшого порядку, ніж q. Поняття ізоморфних та поліморфних функцій розглядалися у навчальному посібнику [48].

Розв'язати проблему ізоморфності перетворення спектрального вектору до множини дискретних значень поля Галуа $GF(q)$ можна через відповідні обмеження, які початково накладаються на елементи спектрального вектору **V** [48].

Такі обмеження безпосередньо випливають з алгебраїчної теорії полів комплексних чисел. Згідно з цією теорією, спектр $P(f)$ має зворотне перетворення Фур'є у полі дійсних чисел тоді та лише тоді, коли виконуються тотожність

$$P^*(f) = P^*(-f),$$

де $P^*(f)$ – комплексно-спряжений вектор до вектору $P(f)$ [63].

Розглянемо відповідну властивість зворотного перетворення Фур'є, яка дозволяє знайти обмеження на значення поля Галуа $GF(q^m)$ згідно із наведеною умовою спряжності.

Властивість 3.9. Якщо **V** є вектором розмірності n в полі Галуа $GF(q^m)$ та n є дільником $q^m - 1$, тоді зворотне перетворення Фур'є від вектору **V** є вектором в полі Галуа $GF(q)$ лише у разі виконання умови:

$$V_j^q = V_{(qj)},\ j = 0, 1, \ldots, n-1. \qquad (3.283)$$

У формулі (3.283), як і в формулі (3.279), запис (qj) означає, що

обчислення проводяться у полі Галуа $GF(q)$. Доведення теореми, пов'язаної із властивістю 3.9, можна знайти у монографії [63].

Для застосування розглянутої властивості зворотного дискретного перетворення Фур'є 3.9 у теорії кодування вводиться поняття про класи спряжених елементів. Надамо відповідне визначення [63].

Визначення 3.16. Спряженими елементами у поля Галуа $GF(q)$ називаються елементи, які обчислюються згідно з наступним співвідношенням:

$$A_j = \{j, jq, jq^2, \dots, jq^{m_j-1}\}, \tag{3.284}$$

де m_j – найменше ціле додатне число, для якого виконується тотожність

$$jq^{m_j-1} = j(\mathrm{mod}\, n). \tag{3.285}$$

Значення m_j, для якого виконується алгебраїчне порівняння (3.285), завжди існує як наслідок скінченності поля Галуа.

Тобто, способи обчислення спряжених компонентів у полях Галуа пов'язані із теорією чисел та методами розв'язування алгебраїчних порівнянь. Відповідний теоретичний матеріал був розглянутий у навчальному посібнику [48].

Часто в теорії чисел для позначення симетрії відповідних елементів спряжені компоненти полів Галуа позначаються від'ємними числами. Оскільки з точки зору циклотомічних класів операції у полі Галуа є зімкненими, від'ємне число j можна заміняти на додатне $n + j$. Теорія циклотомічних класів також розглядалася у навчальному посібнику [48].

Наведемо деякі класи спряжених елементів для полів Галуа невисоких порядків [63].

За модулем 7: {–1, –2, 3}, {0}, {1, 2, –3}.

За модулем 15: {–1, –2, –4, 7}, {0}, {1, 2, 4, –7}, {3, 6, –3, –6}, {5, –5}.

За модулем 21: {–3, –6, 9}, {–1, –2, –4, –8, 5, 10}, {0}, {1, 2, 4, 8, –5, –10}, {3, 6, –9}, {7, –7}.

Із класами спряжених елементів пов'язане поняття сліду, яке розглядатиметься у наступному підрозділі.

3.6.3 Сліди та їхні властивості

Клас спряжених елементів поля виділяє в спектрі множину частот. Надамо відповідні визначення [63].

Визначення 3.17. Множина частот, яка відповідає спряженим елементам, називається хордою.

Визначення 3.18. Слідом порядку q для елемента β у полі Галуа $GF(q)$ називається сума:

$$\mathrm{tr}(\beta) = \sum_{i=0}^{m-1} \beta^{q^i}. \qquad (3.286)$$

Поняття сліду є одним із головних у теорії спектрального аналізу дискретних числових послідовностей, визначених у полі Галуа. Розглянемо головні властивості слідів, які ефективно використовуються для формування групових кодів.

Властивість 3.10. Якщо клас спряжених елементів, до якого належить елемент β, є множиною із кількістю елементів m, тоді слід цього елементу дорівнює сумі класа спряжених елементів.

Дійсно, у противному випадку число елементів ν у класі спряжених є дільником числа m, і тоді кратность входження кожного елементу до сліду tr(β) дорівнює частці m/ν [63].

Наслідком властивості 3.10 є властивість комутативності суми для сліду [63].

Властивість 3.11. Слід від суми двох елементів дорівнює сумі слідів кожного з них, тобто:

$$\mathrm{tr}(\beta + \gamma) = \mathrm{tr}(\beta) + \mathrm{tr}(\gamma). \qquad (3.287)$$

Важливою є також властивість про сліди спряжених елементів [63].

Властивість 3.12. Всі спряжені елементи мають однаковий слід.

Іншою важливою властивістю слідів є властивість про їхню кількість у полі Галуа $GF(q^m)$ [63].

Властивість 3.13. Над полем Галуа $GF(q^m)$ слід порядку q приймає значення кожного з елементів поля Галуа $GF(q)$ однакову кількість разів, яка становить q^{m-1}.

У теорії кодування важливою є також властивість про квадратне рівняння [63].

Властивість 3.14. Квадратне рівняння $x^2 + x + a = 0$, де a – елемент поля Галуа $GF(2^m)$, має розв'язок у цьому полі тоді і лише тоді, коли слід елементу a дорівнює 0.

Доведення відповідної теореми, якій відповідає властивість 3.14, виходить за рамки цього посібника. Його можна знайти у монографії [63].

Визначимо тепер відповідну хорду A_k та опишем її спектр наступним чином:

$$W_j = \begin{cases} 0, & j \in A_k; \\ 1, & j \notin A_k. \end{cases}$$

Згідно із властивістю 3.9 зворотне перетворення Фур'є для цього спектру – це вектор у полі Галуа $GF(q)$. Такий вектор, згідно із властивістю 3.6, можна подати у вигляді поліному $\omega(x)$.

Тепер застосуємо властивість 3.4 про згортку. Зважаючи на те, що в частотній області згортка $\omega^2(x)$ відображується на добуток W_j^2 , а, згідно із (3.287), для будь-яких значень j $W_j^2 = W_j$, для полінома $\omega(x)$ завжди є правильною наступна тотожність [63]:

$$\omega^2(x) = \omega(x)\big(\mathrm{mod}(x^n - 1)\big). \qquad (3.288)$$

Згідно із співвідношенням (3.288), надамо відповідне визначення, яке також є важливим у теорії спектрального аналізу дискретних послідовностей [63].

Визначення 3.19. Довільний поліном $\omega(x)$, для якого виконується порівняння (3.288), називається ідемпотентом.

3.6.4 Спектральне подання циклічних кодів та кодів Ріда – Соломона

Розглянемо спочатку одну із властивостей циклічних кодів пов'язану з визначенням 3.19 та співвідношенням (3.288) [63].

Властивість 3.15. Для кожного циклічного коду існує єдиний поліном $\omega(x)$, який є для нього ідемпотентом.

Тобто, згідно із визначенням 3.19 та властивістю 3.15, для будь-якого циклічного коду виконуєтсья співвідношення [63]:

$$c(x)\omega(x) = c(x)\big(\mathrm{mod}(x^n - 1)\big). \quad (3.289)$$

Згідно із теоретичними відомостями, наведеними у підрозділі 2.5, кожне слово циклічного коду $c(x)$ задається як добуток інформаційного поліному $d(x)$ на твірний поліном $g(x)$, тобто:

$$c(x) = g(x)d(x). \quad (3.290)$$

Тоді, згідно із визначенням циклічного коду та співвідношенням (3.290), у часовій області коефіцієнти циклічного коду c_i являють собою згортку від добутку коефіцієнтів твірного поліному $g(x)$ на інформаційний поліном $d(x)$ [63]:

$$c_i = \sum_{k=0}^{n-1} g_{(i-k)} d_k. \quad (3.291)$$

Тоді в частотній області операція кодування, згідно із властивістю 3.4, засисується як добуток коефіцієнтів відображень інформаційного та твірного поліномів [63]:

$$C_j = G_j D_j. \quad (3.292)$$

Будь-який спектр, який відповідає наведеному співвідношенню (3.292), задає в частотній області кодове слово за умови, що всі алгебраїчні опеарції здійснюються над елементами поля Галуа $GF(q)$.

Розглянемо спочатку приклад побудови в частотній області коду Хеммінга. Для виконання цього завдання використаємо властивість 3.9 та співвідношення (3.283) [63].

Приклад 3.49. Побудувати в частотній області код Хеммінга (7, 4).

Розглянемо коефіцієнти кода Хемінга в частотній області C_0, C_1, C_2, C_3, C_4, C_5 та C_6. Як частоти для перевірки оберемо коефіцієнти C_1 та C_2, за умови такого вибору з використанням сформованого коду можна буде виправити одиночну помилку. Частоти, яким відповідають розряди C_0 та C_3, будемо вважати інформаційними. Тоді решту розрядів можна визначити з використанням властивості 3.9 та співвідношення (3.283). Для лінійного коду

Хеммінга маємо наступний результат:

$$C_1^4 = C_2^2 = C_4 = 0, \qquad C_3^4 = C_6^2 = C_5.$$

З іншого боку, згідно з властивістю 3.9, $C_0^2 = C_0$, і за такої умови коефіцієнт C_0 може приймати лише значення 0 або 1. Можна зробити висновок, що компоненті C_0 відповідає 1 біт, а компоненті C_3 – 3 біти. Тобто, для однозначного визначення спектра необхідно в частотній області задати 4 інформаційних біти кода Хеммінга.

У загальному випадку числа розбиваються на класи спряжених елементів згідно із визначенням 3.16 та формулою (3.284). Із цього можна зробити наступний важливий висновок, який стосується загального принципу формування кодів у спектральній області. Якщо задана спектральна компонента коду C_j, будь-яка інша компонента, індекс якої належить до класу спряжених із j елементів, не може бути обраною довільно, оскільки вона є степінню елемента C_j. Якщо потужність множини класу спряжених елементів у полі Галуа $GF(q)$ дорівнює r, тоді завжди мають місце наступні тотожності:

$$C_j^{q^r} = C_i, \quad C_j^{q^r-1} = 1. \tag{3.293}$$

Відповідні коефіцієнти C_j для коду Хеммінга (7, 4) для різних кодових послідовностей часової області, записані через степені примітивного елементу α, наведені у таблиці 3.22.

Таблиця 3.22 – Коди Хеммінга (7, 4) в часовій області та відповідні коди в частотній області, записані через степені примітивного елемента α у полі Галуа $GF(q)$

Кодові слова у частотній області							Кодові слова у часовій області						
C_0	C_1	C_2	C_3	C_4	C_5	C_6	c_0	c_1	c_2	c_3	c_4	c_5	c_6
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	α^0	0	α^0	α^0	1	1	1	0	1	0	0
0	0	0	α^1	0	α^4	α^2	0	0	1	1	1	0	1

0	0	0	α^2	0	α^1	α^4	0	1	0	0	1	1	1
0	0	0	α^3	0	α^5	α^6	1	1	0	1	0	0	1
0	0	0	α^4	0	α^2	α^1	0	1	1	1	0	1	0
0	0	0	α^5	0	α^6	α^3	1	0	0	1	1	1	0
0	0	0	α^6	0	α^3	α^5	1	0	1	0	0	1	1
1	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	α^0	0	α^0	α^0	0	0	0	1	0	1	1
1	0	0	α^1	0	α^4	α^2	1	1	0	0	0	1	0
1	0	0	α^2	0	α^1	α^4	1	0	1	1	0	0	0
1	0	0	α^3	0	α^5	α^6	0	0	1	0	1	1	0
1	0	0	α^4	0	α^2	α^1	1	0	0	0	1	0	1
1	0	0	α^5	0	α^6	α^3	0	1	1	0	0	0	1
1	0	0	α^6	0	α^3	α^5	0	1	0	1	1	0	0

Із наведених вище теоретичних відомостей зрозуміло, що довільний елемент із поля Галуа $GF(q^m)$ не може бути значенням коефіцієнту C_j. Припустимими є або нульовий елемент, або ті елементи, порядок яких є дільником величини $q^r - 1$ [83]. Також зрозуміло, що потужність кожного із класів спряжених елементів є дільником порядку поля m.

Для описання принципу роботи частотного кодера необхідно розбити множину із $q^m - 1$ чисел на класи спряжених елементів та обрати одного з представників цього класу. Ці представники однозначно визначають значення інформаційних символів.

Тобто, структура циклічного коду в частотній області має наступний вигляд. Спочатку обераються $2t$ спектральних компонент, які вважаються нульовими. Решта визначених символів є інформаційними та можуть приймати довільні значення згідно із обмеженнями, обумовленими порядком поля Галуа. І нарешті третій набір символів, індекси яких належать до класу

спряжених елементів, формує зв'язані частоти. Ці символи не є довільними, їх значення визначається структурою коду [83].

У таблиці 3.23 наведені набори зв'язаних частот для поля Галуа *GF*(64).

Таблиця 3.23 – Структура частотного спектру для поля Галуа *GF*(64)

Вільні частоти	Зв'язані частоти	Кількість бітів
C_0	–	1
C_1	$\{C_2, C_4, C_8, C_{16}, C_{32}\}$	6
C_3	$\{C_6, C_{12}, C_{24}, C_{48}, C_{33}\}$	6
C_5	$\{C_{10}, C_{20}, C_{40}, C_{17}, C_{34}\}$	6
C_7	$\{C_{14}, C_{28}, C_{56}, C_{49}, C_{35}\}$	6
C_9	$\{C_{18}, C_{36}\}$	3
C_{11}	$\{C_{22}, C_{44}, C_{25}, C_{50}, C_{37}\}$	6
C_{13}	$\{C_{26}, C_{52}, C_{41}, C_{19}, C_{38}\}$	6
C_{15}	$\{C_{30}, C_{60}, C_{57}, C_{51}, C_{39}\}$	6
C_{21}	$\{C_{42}\}$	2
C_{23}	$\{C_{49}, C_{29}, C_{58}, C_{53}, C_{43}\}$	6
C_{27}	$\{C_{54}, C_{45}\}$	3
C_{31}	$\{C_{62}, C_{61}, C_{59}, C_{55}, C_{47}\}$	6

3.6.5 Розширені коди Ріда – Соломона

До коду Ріда – Соломона, розглянутого у підрозділі 3.4, у загальному випадку можна добавити дві додаткові компоненти, які зазвичай розташовуються на початку та наприкінці кодового слова. Коди, які отримуються таким способом, в літературі називаються розширеними кодами Ріда – Соломона [83].

Слід відзначити, що кожен з двох символів, які додаються до коду, може розглядатися і як контрольний, і як інформаційний. Це надає можливість реалізувати два альтернативних підходи. У разі збільшення кількості

інформаційних символів підвищується швидкість передавання інформації, а у разі збільшення кількості контрольних символів збільшується кодова відстань та, відповідно, завадостійкість коду.

Для побудови розширеного коду Ріда – Соломона необхідно визначити два нових локатора, та, відповідно, ввести нові позначення. Якщо початкові компоненти нумеруються елементами поля, тоді для однієї з двох введених додаткових компонент є можливість використати нульовий елемент, у цьому разі залишається лише визначити номер другої компоненти. Зазвичай для неї використовується символ нескінченності (∞) [83]. Але якщо для нумерації компонентів початкового коду використовуються степені примітивного елемента, тоді номер 0 вже зайнятий, і у цьому випадку слід ввести два нових символи. Якщо ми скористаємося для цих двох символів знаками – та +, кодове слово розширеного коду Ріда – Соломона можна записати у вигляді:

$$\left(c_{-}, c_0, c_1, c_2, \dots, c_{q^m-3}, c_{q^m-2}, c_{+}\right). \tag{3.294}$$

У записаному співвідношенні (3.294) кількість символів коду n становить [83]:

$$n = q^m + 1. \tag{3.295}$$

Надамо відповідні визначення [83].

Визначення 3.20. Вектор, який створюється як результат викресленя з розширеного коду Ріда – Соломона (3.294) першого та останнього символів c_{-} та c_{+}, називається внутрішнім вектором кодової комбінації (англійський термін – internal codeword vector).

У разі розбиття розширеного коду Ріда – Соломона на внутрішню кодову комбінацію та додаткові символи, властивості розширених кодів Ріда – Соломона можна вивчати з використанням властивостей перетворення Фур'є, відповідний теоретичний матеріал був розглянутий у першій частині посібника [1] та у першому розділі цієї частини посібника.

Визначення 3.21. Розширеним циклічним кодом (n, k) над полем Галуа

$GF(q)$ називається лінійний код із довжиною $n = q^m + 1$, кожне кодове слово якого задовольняє наступним двом умовам. Спектр (C_0, C_0, …, C_{n-3}) кодового слова (c_-, c_0, c_0, …, c_{n-3}, c_+) містить нулі на заданій множині $n - k - 2$ позицій з індексами j_1, …, j_{n-k-2}, а дві спектральні компоненти, які залишаються, мають значення $C_{j0} = c_-$, $C_{j_{n-k-1}} = c_+$ [83].

Слід відзначити, що розширений циклічний код не завжди відповідає властивостям циклічних кодів, які були описані у підрозділі 2.5.

На основі визначення 3.21 надамо визначення розширеного коду Ріда – Соломона [83].

Визначення 3.22. Розширеним кодом Ріда – Соломона називають лінійний код довжиною $n = q + 1$ над полем Галуа $GF(q)$, спектр кодових слів якого (c_-, c_0, c_0, …, c_{n-2}, c_+) для будь-яких цілих значень j_0 та t задовольняє наступним умовам: 1) $C_j = 0$; $j = j_0 + 1, \ldots, j_0 + 2t - 2$; 2) $C_{j0} = c_-$; 3) $C_{j0+2t-1} = c_+$.

Головною властивістю розширеного коду Ріда – Соломона є те, що число $2t + 1$ завжди є його конструктивною кодовою відстанню. Визначення 3.22 містить обмеження, яке полягає у тому, що $2t - 2$ спектральних компонент дорівнюють 0, а крайня діва та крайня права компоненти визначеної кодової послідовності, відповідно дорівнюють c_- та c_+. Ці дві компоненти являють собою граничні частоти. Порівняно із звичайним кодом Ріда – Соломона розширений код завжди містить два додаткових інформаційних символи, проте його кодова відстань за цієї умови не змінюється [83].

Структурна схема кодера, призначеного для створення розширеного коду Ріда – Соломона, наведена на рис. 3.56.

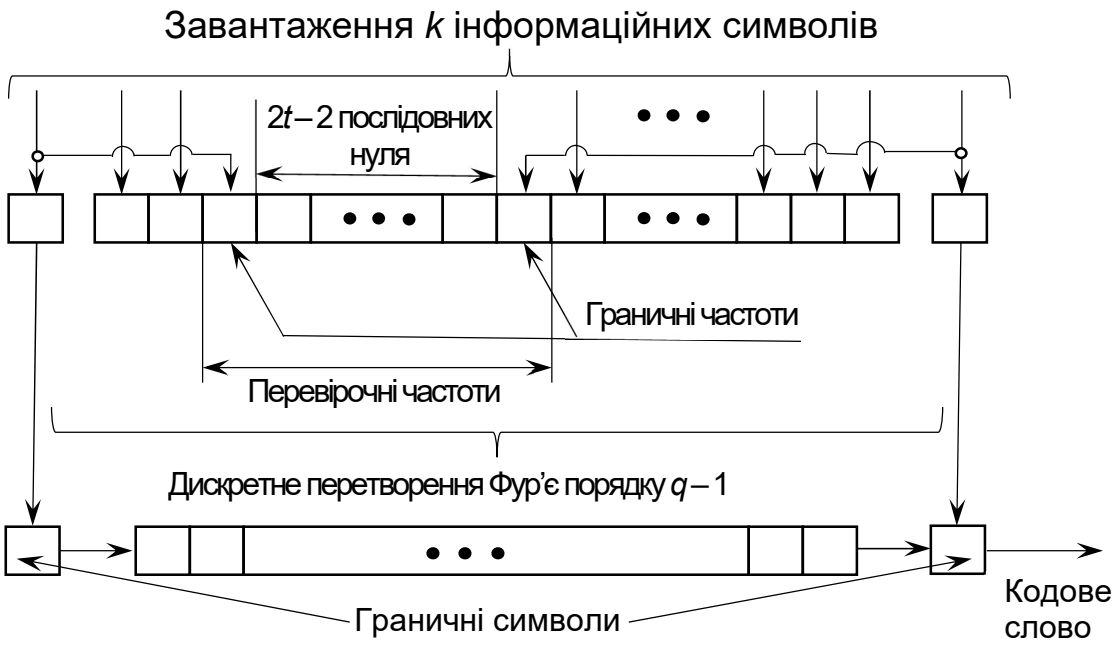


Рис. 3.56 Узагальнена структурна схема кодера, призначеного для формування розширеного коду Ріда – Соломона в частотній області

У разі, якщо стоїть завдання розширити початкову послідовність коду Ріда – Соломона так, щоб збільшити його мінімальну кодову відстань, тоді два додаткових символи, які приєднуються до блоку перевірочних частот, визначаються як нові перевірочні частоти. Значення компонентів цих частот не змінюються, але у часовій області ті ж самі два символи дописуються до кодового слова. Такий код має однакову кількість інформаційних символів відносно до початкового коду, але його мінімальна кодова відстань становить $d^* + 2$, де d^* – кодова відстань початкового коду. Якщо довжина послідовності початкового коду складає n, а розширеного – n', тоді граничні символи розширеного коду Ріда – Соломона визначаються згідно із наступними співвідношеннями:

$$c_- = \sum_{i=0}^{n'-1} c_i \alpha^j, \qquad c_+ = \sum_{i=0}^{n'-1} c_i \alpha^{2ti}, \tag{3.296}$$

де α – примітивний елемент поля, α^2, …, α^{2t-1} – корені породжувального поліному. Слід відзначити, що аналогічний результат можна отримати, якщо використати твірний поліном (3.199) та алгоритм, заданий співвідношеннями

(3.219) за умови $t' = t + 2$, де t – кількість помилок, які виправляє початковий код, t' – кількість помилок, які виправляє розширений код [83].

3.7 Ітеративні коди

Ітеративні коди, як і прямокутний код, розглянутий у підрозділі 2.3.3, та каскадні коди, що розглядалися у підрозділі 3.5, будуються на алгоритмах матричного аналізу. Відповідні теоретичні відомості щодо матричних методів формування кодових послідовностей були розглянуті у першому томі другої частини цього посібника [48].

Взагалі для ітеративних кодів характерним є то, що операції кодування здійснюються над сукупностями символів, які розташовуються за різними координатами [5]. Припустимо, що кількість таких координат становить q. За такої умови загальна кількість інформаційних символів ітеративного коду складає:

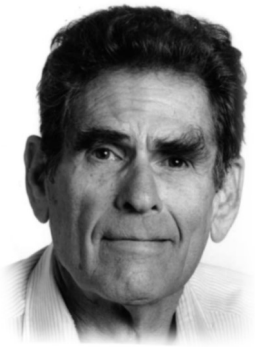
$$m = \sum_{\gamma=1}^{q} m_{\gamma}, \tag{3.296}$$

де m_γ – кількість інформаційних символів за координатою γ.

В ітеративному коді послідовності інформаційних символів за кожною з координат кодуються відповідним лінійним кодом, тому у загальному випадку кожний інформаційний символ входить до q кодових векторів [5].

Вперше ідея створення класичних ітеративних кодів була запропонована американським математиком П. Елайєсом [85]. У найпростішому ітеративному коді із розмірністю $q = 2$ формується матриця із l векторів однакової розмірності n. Відповідна структура ітеративного коду показана на рис. 3.57. Зрозуміло, що з точки зору способу кодування каскадний та прямокутний коди також є ітеративними.

Рис. 3.57 Узагальнена структура ітераційного коду



Пітер Елайєс (1923 – 2001)

Як видно з рис. 3.57, контрольні символи в ітеративному коді розташовані у стовпчику n та у рядку l. Для простого варіанту лінійного ітеративного коду ці символи розраховуються за співвідношеннями [5]:

$$a_{jn} = \sum_{i=1}^{n-1} a_{ij} (\text{mod} 2);\ a_{li} = \sum_{j=1}^{l-1} a_{ji} (\text{mod} 2);\ \ a_{ln} = \sum_{j=1}^{l-1} a_{jn} (\text{mod} 2). \quad (3.297)$$

Передавання символів ітераційного коду зазвичай здійснюють або послідовно, символ за символом, або паралельно, цілими рядками. Перевірка виконання співвідношень (3.297) дозволяє перевірити непарну кількість спотворених символів. Тобто, невиявленими залишаються лише помилки, яким відповідає парна кількість спотворених символів як за рядками, так і за стовпчиками. Найпростіша помилка, яку не можна виявити з використанням ітеративного коду, є помилка з чотирьох спотворених символів, які

розташовані у вершинах прямокутника. Відповідна ситуація відображена на рис. 3.58.

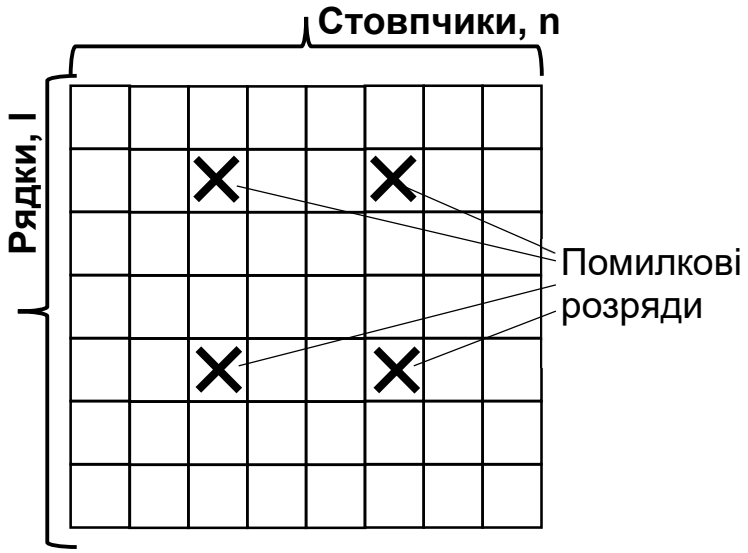


Рис. 3.58 Ілюстрація помилки із чотирма спотвореними символами, яка не виявляється ітеративним кодом

Кількість помилок із прямокутним розташуванням, аналогічних помилці, показаній на рис. 3.58, згідно із законами комбінаторики, описаними у шостому розділі другої частини посібника [49], становить:

$$B_{4\text{п}} = C_n^2 C_l^2 = \frac{nl(n-1)(l-1)}{2!2!} \tag{3.298}$$

за умови загальної кількості чотириразових помилок:

$$B_{4\text{з}} = C_{nl}^2 = \frac{nl(nl-1)(nl-2)(nl-3)}{4!}. \tag{3.299}$$

Виходячи із записаних співвідношень (3.298), (3.299), можна записати остаточне співвідношення для відносної кількості чотириразових помилок, які не виправляє ітеративний код, до загальної кількості таких помилок, у такому вигляді [5]:

$$\frac{B_{4\text{п}}}{B_{4\text{з}}} = \frac{6!(n-1)(l-1)}{(nl-\ )(nl-2)(nl-3)}. \tag{3.300}$$

Визначимо мінімальну вагу ненульового вектору для двовимірного ітеративного коду із матричним поданням, який розглядається. Такий вектор

повинен містити лише один ненульовий вектор-рядок, мінімальна вага якого становить 2. Перевірка на парність кожного з ненульових стовпчиків також дає вектор із вагою 2. Тоді мінімальна вага ненульового вектору простого двовимірного ітеративного коду із подвійною перевіркою на парність дорівнює 2·2 = 4 [5].

Аналогічно можна показати, що для ітеративного коду із розмірністю q мінімальна вага ненульового вектору обчислюється наступним чином [5]:

$$d = \sum_{\gamma=1}^{q} d_{\gamma}, \qquad (3.301)$$

де d_{γ} – кодова відстань лінійного коду за координатою γ.

Корекція помилок в ітеративному коді здійснюється послідовно, спочатку за першою координатою, потім – за другою, і так далі. Така процедура декодування є досить простою, але її недолік полягає у тому, що знижається коректувальна здатність коду, оскільки частини помилок із кратністю $\frac{d-1}{2}$ стає неможливим.

Розглянемо відповідний приклад, у якому оцінимо коректувальну здатність ітеративного коду, побудованого на основі коду Хеммінга.

Приклад 3.50. Побудувати ітеративний код на основі коду Хеммінга (7,4) та оцінити його коректувальну здатність.

Якщо кодування всіх рядків здійснюється через код Хеммінга (7,4), матриця ітеративного коду буде мати розмірність (49, 16). З одного боку, такий ітеративний код, згідно із співвідношенням (3.301), має мінімальну кодову відстань $d = 3 \cdot 3 = 9$. Тобто, лінійний код із аналогічною структурою буде виправляти всі помилки, кратність яких становить 4 або є нижчою. Але, як було показано вище, ітеративний код не може виправляти чотириразові помилки, якщо вони розташовані у вершинах прямокутника, тобто, якщо ці помилки виникли у двох однакових стовпчиках та рядках. Тобто, коректувальна здатність побудованого ітеративного коду відповідає виправленню трьох помилок, які виникають одночасно.

Таким чином, лінійні коди з відповідною кодовою відстанню є більш ефективними з точки зору можливості корекції відповідної кількості помилок, але несумнівною перевагою ітеративного коду є відносна простота його

побудови [5]. Зрозуміло також, що ітеративні коди з інформаційної точки зору є надлишковими.

Двоступеневі ітеративні коди знайшли широке впровадження в системах записування інформації на магнітні стрічки та на магнітні диски [5]. Структура таких кодів базується на аналізі статистики помилок. Така статистика показує, що під час експлуатації магнітних носіїв переважно виникають помилки вздовж фізично пошкоджених секторів доріжок, а імовірність виникнення дух пачок помилок на різних доріжках в одному інформаційному кадрі є вкрай низькою. Розглянемо один із прикладів стандартного ітеративного коду, призначеного для надійного запису цифрової інформації на магнітну стрічку.

Приклад 3.51. Спочатку розглянемо узагальнену структуру цього ітеративного коду. У кожному інформаційному кадрі формується додатковий рядок, а перевірочні символи на додатковій доріжці визначаються згідно з умовою забезпечення наявності непарної кількості одиниць вздовж заданого рядка. А в напрямку читання доріжок формується паралельний циклічний код, що дозволяє точно визначати номер доріжки, на якій виникла пачка помилок. Тоді точна координата помилки визначається двома значеннями: номером доріжки, на який вона виникла, та результатом перевірки на парність за рядками. Відповідний спосіб пошуку помилки показаний на рис. 3.59.

Рис. 3.59 Наочна ілюстрація способу формування ітеративного коду, призначеного для надійного запису цифрової інформації на магнітну стрічку

Більш досконало структура ітеративного коду, призначеного для надійного

запису цифрової інформації на магнітну стрічку, показана у таблиці 3.24 [5].

Таблиця 3.24 – Структура ітеративного коду, призначеного для запису цифрової інформації на магнітну стрічку

Інформаційні доріжки						Доріжка доповнення до непарності
1	2	. . .	j	. . .	$n-1$	n
$F_1(x)a_{0,1}$	$a_{1,1}x^1$	. . .	$a_{j-1,1}x^{j-1}$	. . .	$a_{n-2,1}x^{n-2}$	$a_{n-1,1}x^{n-1}$
$F_2(x)a_{0,2}$	$a_{1,2}x^1$	. . .	$a_{j-1,2}x^{j-1}$	. . .	$a_{n-2,2}x^{n-2}$	$a_{n-1,2}x^{n-1}$
. . .	. . .	. . .	. . .	. . .	. . .	. . .
$F_t(x)a_{0,t}$	$a_{1,t}x^1$	. . .	$a_{j-1,t}x^{j-1}$	. . .	$a_{n-2,t}x^{n-2}$	$a_{n-1,t}x^{n-1}$
$R_t(x)a_{0,t+1}$	$a_{1,t+1}x^1$	. . .	$a_{j-1,t+1}x^{j-1}$	. . .	$a_{n-2,t+1}x^{n-2}$	$a_{n-1,t+1}x^{n-1}$

Кодування кадрів циклічним кодом в ітеративному коді, призначеному для запису цифрової інформації на магнітну стрічку, здійснюється наступним чином [5]. Поліном $F_1(x)$, який відповідає інформації, записаній у першому рядку, множиться на змінну x та результат зводиться за модулем твірного поліному степені n $g(x)$, де n – загальна кількість доріжок, включаючи контрольну. Наприклад, для дев'яти доріжок $g(x) = (x + 1)(x^8 + x^4 + x^3 + x^2 + 1)$ [5]. Отримана остача $R_1(x)$ сумується за модулем 2 з поліномом $F_1(x)$, який відповідає інформації, записаній у другому рядку. Отримана сума знову множиться на x та зводиться за модулем твірного поліному $g(x)$, в результаті здійснення цієї операції визначається нова остача $R_2(x)$. Цей алгоритм ітераційно застосовується до всіх рядків інформаційного кадру, який кодується, в результаті цієї операції формується поліном $R_l(x)$, який записується в кінці кадру. Цей поліном має наступний вигляд [5]:

$$x^l F_1(x) + x^{l-1} F_2(x) + \ldots + xF_l(x) = R_l(x) \bmod g(x). \qquad (3.302)$$

Аналогічним чином здійснюється декодування інформації. Визначимо поліном $M(x)$ як [5]:

$$M(x) = x^{l+1}F_1(x) + x^lF_2(x) + \ldots + x^2F_l(x) + xR_l(x). \quad (3.303)$$

Тоді у разі наявності помилки $E(x)$ має місце тотожність [5]:

$$M(x) + E(x) = R'(x) \bmod g(x)\,, \quad (3.304)$$

а у разі відсутності помилки буде зафіксоване значення $R'(x) = 0$. У такий спосіб циклічний код може виправити пачку помилок, якщо вони виникли на одній доріжці. Дійсно, якщо $R'(x) \neq 0$, поліном $E(x)$ визначається як [5]:

$$E(x) = x^j e(x), \quad (3.305)$$

де $e(x)$ – порозрядний вектор помилки.

Зрозуміло, що описаний спосіб кодування, використаний в ітеративному коді, призначеному для записування інформації на магнітну стрічку, цілком відповідає алгоритму створення систематичних циклічних кодів, який був описаний у підрозділі 2.5.5.

Номер доріжки із спотвореними символами визначається із наступних міркувань. Припустимо, що під час читання на доріжці із номером n виникла пачка помилок із вектором $e(x)$. Тоді результатом декодування буде поліном $R''(x)$, який записується наступним чином [5]:

$$R''(x) = x^{j+k}e(x)\mathrm{mod}\big(g(x)\big), \quad (3.306)$$

де k – кількість доріжок, на яку відстає доріжка з відомим номером n від доріжки, де виникли помилки. Тоді, для декодування інформації, необхідно визначити дві остачі – $R'(x)$ та $R''(x)$, Множення поліному $R'(x)$ на змінну x^w, де $w = 1, 2, \ldots, n$, дозволяє знайти шукане значення k, оскільки [5]:

$$R''(x) = x^k R'(x). \quad (3.307)$$

Якщо описана ітераційна процедура не дає результату та співвідношення (3.307) не виконується, вважається, що виникла помилка, яку не можна виправити [5].

Ітеративні коди досить часто використовувались на практиці на початку

розвитку інформаційних технологій для кодування та декодування невеликої кількості інформації, але суттєвим їхнім недоліком є досить велика надлишковість. Починаючи з вісімдесятих років XX століття для кодування інформації на магнітних та оптичних носіях почали все частіше використовувати групові коди Ріда – Соломона, які мають кращу коректувальну здатність. Така зміна способу завадостійкого кодування пов'язана з тим, що для кодів Ріда – Соломона були розроблені ефективні ітераційні алгоритми декодування, описані у підрозділі 3.4. У підрозділі 3.8.11.2 розглядатимуться згорткові коди, призначені для кодування інформації на магнітних та оптичних носіях.

3.8 Згорткові коди

3.8.1 Загальний принцип формування згорткових кодів

Згортковий код (англійський термін – convolutional code) суттєво відрізняється від лінійних та блокових кодів, які розглядалися раніше, насамперед тим, що він будується зовсім іншим способом. Особливістю конструкцій лінійних кодів, які розглядалися у розділі 2 та у підрозділах 3.2, 3.3 та 3.4, є те, що їх коректувальна здатність повністю описується двома цілими числами, а саме: загальною кількістю символів n та кількістю інформаційних символів k. Для систематичних лінійних кодів саме ці параметри завжди визначають коректувальну здатність коду. Відповідно до цих параметрів також будуються алгоритми декодування як двійкових, так і групових кодів. Як було показано вище, ці алгоритми базуються на обчисленні контрольних сум, на використанні теорії двійкових поліномів, на матричному аналізі, а також на теорії дискретних груп Галуа. Відповідний математичний апарат розглядався у першому томі другої частини посібника [48]. Загальною відмінною рисою всіх цих алгоритмів є точність математичного подання, і в результаті цього значно зменшується імовірність помилок під час передавання даних. Але такі

точні математичні алгоритми мають і свої суттєві недоліки, головний із яких – підвищення надлишковості інформації, яка передається через канал зв'язку. Ця надлишковість пов'язана із тим, що кількість символів n, які передаються, завжди є однаковою, незалежно від того, наскільки важливою є для приймальної сторони інформація, яка передається. Відповідні методи теоретичних оцінок кількісті інформації, яка передається через канал зв'язку, розглядалися у першій частині посібника [1].

Алгоритми роботи згорткових кодів є дещо іншими. У цих кодах використовуються попередні інформаційні оцінки, які зазвичай базуються на методах теорії імовірностей, теорії інформації та систем штучного інтелекту, відповідний математичний апарат розглядався у другому та третьому томах другої частини посібника [49, 50]. Зокрема, для оцінок коректувальної здатності завадостійких кодів використовують математичний апарат деревоподібних та ґраткових діаграм [48, 50].

У загальному випадку цифрові електронні пристрої для формування згорткових кодів та для декодування їх послідовностей з теоретичної точки зору можна розглядати як скінченні автомати із фіксованою кількістю комірок пам'яті [33]. Сутність роботи цих пристроїв полягає у тому, що символ, який формується на виході декодера, генерується на основі відомих законів теорії імовірностей та теорії інформації [1, 48] з урахуванням попередніх прийнятих символів. Тобто, поточний стан скінченного автомату, в даному випадку, залежить не лише від останнього прийнятого символу, але й від кількох попередніх станів системи. Відповідні положення теорії скінченних автоматів, як одного з найважливіших розділів сучасної дискретної математики, розглядалися у третьому томі другої частини посібника [50].

Головний принцип побудови згорткових кодів, з точки зору теорії множин, полягає у тому, що послідовність символів, які передаються, $\mathbf{m} = m_1, m_2, \ldots m_i$ перетворюється на виході кодеру до вихідної послідовності кодових слів $\mathbf{U} = G(\mathbf{m})$. Тут через i позначений індекс часу для послідовності символів, що передається. Надалі будемо вважати, що символи, які передаються, є

незалежними. Як і для лінійних кодів, відображення групи символів **m** на групу **U** є рівнозначним, проте відмінність згорткового коду полягає у тому, що вихідний вектор **U** визначається не лише послідовністю переданих символів **m**. У згортковому коді аналізується значення всіх попередніх прийнятих символів, і за рахунок цього вихідна послідовність може бути сформована заздалегідь, перед тим, як буде переданий останній символ вхідного повідомлення.

Проте, попре складність використовуваного математичного апарату, базовий принцип побудови згорткових кодів є досить простим та інтуїтивно зрозумілим. Він полягає у тому, що якщо існують попередні дані про інформацію, яка передається, передавати її повністю зазвичай немає необхідності. Повідомлення лише починається – а людина вже може інтуїтивно зрозуміти, про що саме йде мова.

Наведемо з цього приводу доречні приклади. Припустимо, що Ви читаєте підручник з теорії кодування та зустрічаєте там словосполучення Код Хеммінга. Проте проблема полягає у тому, що частина рядка колись була залита чорнилами і замість цієї фрази Ви можете прочитати лише К******мінга. Не зважаючи на те, що частина інформації, яку автор намагався Вам передати, відсутня, Ви легко можете зрозуміти, про що йде мова. А тепер припустимо, що виникла така ж сама ситуація, але читаєте Ви не підручник, а детективний роман. У цьому романі головна дієва особа – це детектив Хеммінг, а в нього на роботі живе дуже розумний кіт, який завжди допомагає йому розкривати злочини. Тоді зрозуміло, що імовірніше за все непрочитаною фразою, яка залита чорнилами, є не Код Хеммінга, а саме Кіт Хеммінга. Інший доречний приклад – це прослуховування не відфільтрованих, зашумлених записів старих пісень відомих виконавців. Деякі слова в них зазвичай не можна розібрати, проте завжди легко зрозуміти зміст пісні, а іноді можна навіть здогадатися, яке саме слово звучить нерозбірливо. Зрозуміло, що такі оцінки інформації є не точними, а імовірнісними, проте зазвичай за контекстом із імовірністю більшою за 99% можна сказати, що спотворене слово, яке не можна прочитати

або почути, є саме таким. Якщо все повідомлення є дуже нерозбірливим – імовірність вгадати його буде значно меншою. Тоді можливі помилки, і тут багато залежить від того, як людина вміє сприймати та аналізувати спотворену інформацію. Дуже багато таких цікавих прикладів є в класичній літератури. Наприклад – це лист капітана Гранта із відомого роману Ж. Верна та передсмертні часи діда Наталки Ростової у романі Л.М. Толстого «Війна і мир».

Для пошуку правильної кодової комбінації за початковими символами повідомлення, яке передається, у згорткових кодах зазвичай використовують статистичний критерій правдоподібності [9, 25, 27, 49]. Цей критерій також відповідає життєвій практиці та ми часто використовуємо його для оцінки достовірності інформаційних повідомлень у різних життєвих ситуаціях. Наприклад, якщо всі студенти в групі знають, що студент Пухленко навчається не дуже добре, а студент Прокопенко майже завжди отримує відмінні оцінки, вони будуть дуже здивовані, отримавши такі повідомлення, як «Пухленко отримав оцінку відмінно» або «Прокопенко отримав оцінку задовільно». Вони вважатимуть ці повідомлення недостовірними, оскільки у випадку, який розглядається, вони не відповідають статистичному критерію правдоподібності.

Розглянемо тепер головні параметри згорткових кодів.

3.8.2 Параметри згорткових кодів та структурна схема кодеру

Лінійні та групові завадостійкі коди, які розглядалися у другому розділі цієї частини посібника та у підрозділах 3.2, 3.3 та 3.4, характеризуються двома головними параметрами: довжиною кодової комбінації n та кількістю інформаційних символів k. На відміну від цього, згорткові коди завжди характеризуються не двома, а трьома параметрами [33].

Параметри n та k також описують коректувальну здатність згорткового коду, але визначаються вони дещо інакше. Параметр k можна розглядати як довжину в бітах вхідного символу згорткового коду, а параметр n – як довжину

в бітах вихідного символу того ж самого коду. Припустимо, що послідовність символів **m** згортковим кодом відображається на вихідну послідовність **U**. Тоді, згідно з розглянутою моделлю подання інформації, послідовність **m** складається з елементарних повідомлень із довжиною *k* бітів, а послідовність **U** – з елементарних повідомлень із довжиною *n* бітів. Щодо довжини послідовностей **m** та **U** – для згорткового коду вони можуть бути різними, незважаючи на те, що функція **U**(**m**) є однозначною. Річ у тому, що в процесі передавання символів вхідної послідовності **m** декодер може вже достроково сформувати вихідне повідомлення, не аналізуючи символи вхідного повідомлення до кінця.

Одним з параметрів згорткового коду також є ціле число *K*, яке називається довжиною кодового обмеження (англійський термін – constraint length). Головна особливість апаратної реалізації кодерів та декодерів згорткових кодів полягає у тому, що, хоча вони завжди мають пам'ять фіксованого об'єму на *n* символів, проте об'єм пам'яті, яка використовується, у значній степені залежить не лише від довжини кодового слова, але і від стану $K-1$ комірок пам'яті, у яких зберігається попередні символи, що були передані. Тобто, декодер згорткового коду дійсно працює як скінченний автомат, а кількість станів, які запам'ятовуються автоматом, становить $K-1$ [33].

Розглянемо спрощену схему кодера згорткового коду, яка наведена на рис. 3.60 [33].

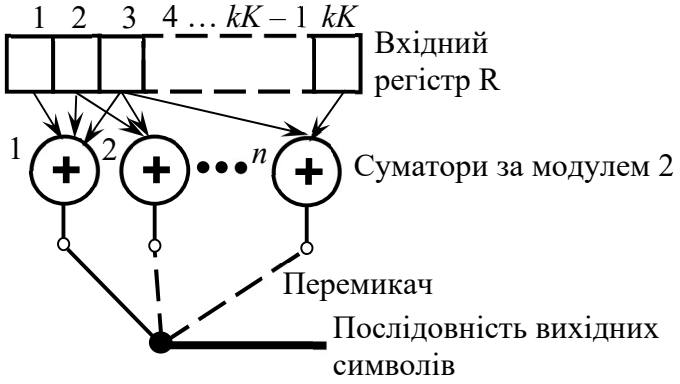


Рис. 3.60 Структурна схема кодеру згорткового коду [33]

Як видно з структурної схеми, наведеної на рис. 3.60, кодер згорткового коду містить *n* суматорів за модулем 2 та вхідний регістр зсуву із кількістю розрядів *kK*. Кодер працює наступним чином. У відповідні моменти часу до

регістру R надходить k бітів вхідної послідовності **m** та всі біти у цьому регістрі зсуються на k позицій праворуч. Синхронно, у ті ж самі моменти часу, всі kK бітів надходять на n суматорів, в результаті чого формуються біти згорткового коду. У системі зв'язку закодоване повідомлення надходить на вхід модулятора, на виході якого формуються двійкові сигнали, що передаються через канал зв'язку. Узагальнена структура цифрової системи зв'язку розглядалася у першій частині посібника [33].

Для спрощення розуміння загального принципу формування згорткових кодів надалі будемо розглядати лише двійкові коди, для яких значення $k = 1$, що з фізичної точки зору означає, що у кодувальному пристрої біти повідомлення, яке надходить, зсуються за одиницю часу на 1 позицію. Загалом розгляд згорткових кодів для більших значень k також можливий та не викликає ніяких труднощів з теоретичної точки зору. Згідно із схемою кодера, наведеною на рис. 3.60, на кожний вхідний біт коду формується n символів вихідного повідомлення. За такої умови степінь кодування, або коефіцієнт надлишковості коду, складає $\frac{1}{n}$ [33]. З бітів повідомлення, які формуються у момент часу t_i, складається слово кодової послідовності $U_i = \{u_{1i}, u_{2i}, \ldots, u_{ji}, \ldots, u_{ni}\}$, де $j = 1, 2, \ldots, n$ – номер символу коду, який відповідає часу t_i. Зрозуміло також, що оскільки вважається, що $k = 1$, довжина ригістру зсуву складає K бітів. Величина K також є обмеженням на довжину кодового слова у бітах.

Розглянемо тепер прості приклади формування згорткового коду та способи подання його кодових послідовностей.

3.8.3 Приклади формування згорткового коду

Розглянемо спочатку принцип роботи простого кодеру, який формує згортковий код із довжиною кодового обмеження $K = 3$ та з коефіцієнтом надлишковості $\frac{1}{2}$. Такі параметри коду означають, що регістр зсуву має 3 розряди, а кількість суматорів зо мадулем 2 становить 2. Різниця між дома суматорами у даному випадку полягає у тому, що перший суматор сумує три

символи, які надходять до вхідного регістра, а другий – лише два, перший та останній символи. Структурна схема такого кодеру наведена на рис. 3.61 [33]. В результаті виконання таких операцій сумування формуються пари бітів вихідних кодових символів, а загалом результати сумування залежать від вмісту вхідного регістру. Під час надходження до вхідного регістру наступного біту здійснюється зсув послідовності символів на одну позицію праворуч, відповідно один із прийнятих бітів втрачається.

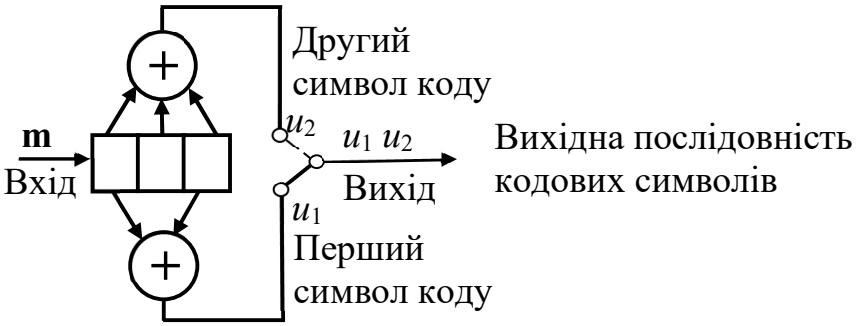


Рис. 3.61 Принцип роботи схеми, яка формує простий згортковий код з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$

Розглянемо приклади роботи такої схеми за умови заданої вхідної кодової послідовності.

Приклад 3.52. Проаналізувати результати роботи кодеру, схема якого наведена на рис. 3.61, у різні моменти часу, вважаючи, що на його вхід подана послідовність бітів 1 0 1.

Будемо аналізувати роботу схеми послідовно.

На початку роботи схеми вхідний регістр заповнений нулями, тому на виході схеми формуються два нуля. У момент часу $t = t_1$ перший символ повідомлення становить $m_1 = 1$, в результаті чого перший біт регістру набирає значення 1, а два інших біти, як і раніше, дорівнюють 0. У цьому разі на виході першого суматора отримуємо значення $u_1 = 1 \oplus 0 \oplus 0 = 1$, а на виході другого – $u_2 = 1 \oplus 0 = 1$.

У наступний момент часу $t = t_2$ надходить другий символ повідомлення $m_2 = 0$, і це

значення завантажується в перший біт вхідного регістру, а решта значень зсується у регістрі на 1 біт праворуч із відкиданням останнього значення. В результаті цієї операції вміст регістру змінюється на значення 0 1 0. За такої умови на виході першого суматора отримуємо значення $u_1 = 0 \oplus 1 \oplus 0 = 1$, а на виході другого – $u_2 = 0 \oplus 0 = 0$.

Відповідні результати роботи схеми у різні моменти часу показані на рис. 3.62.

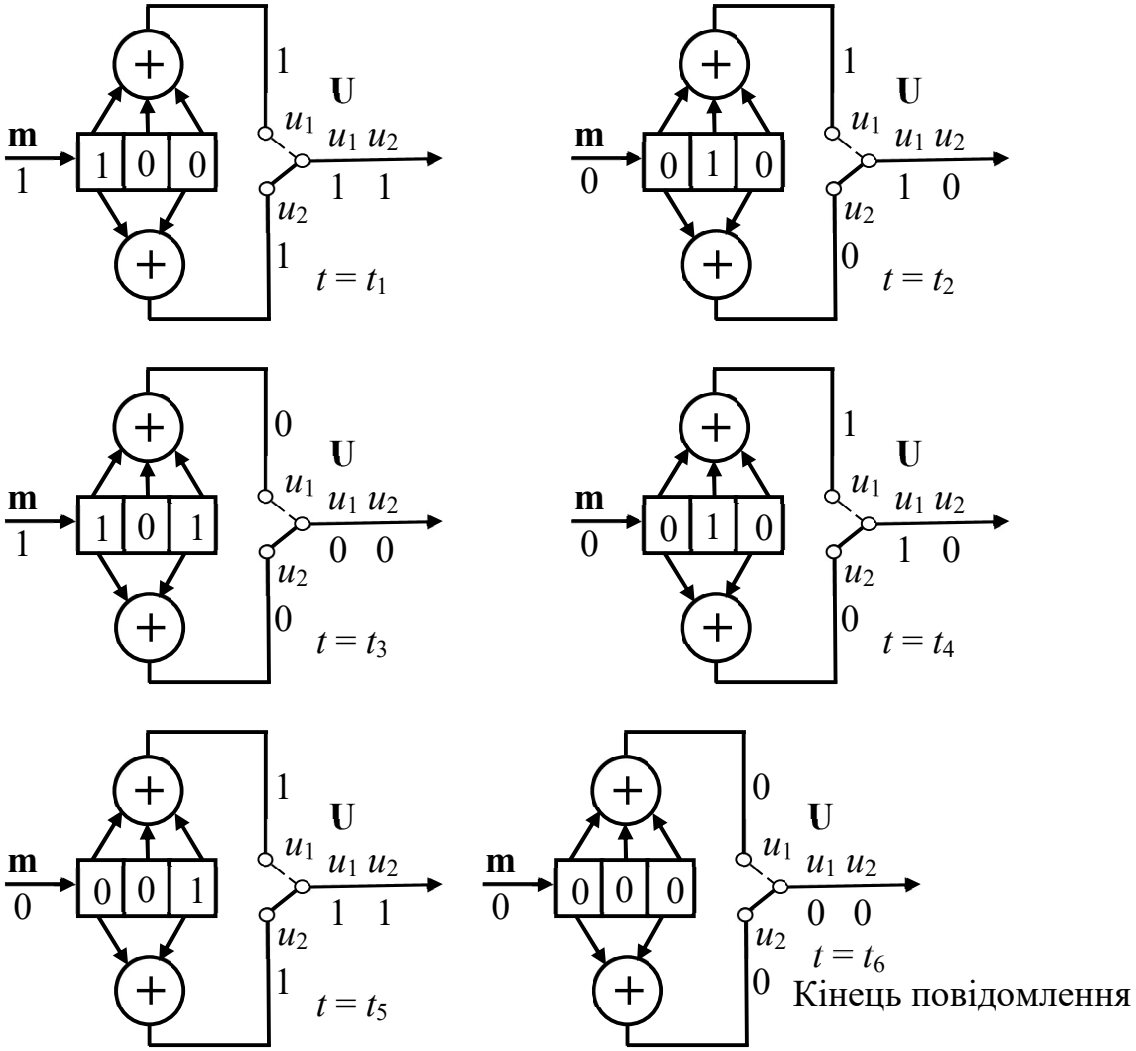


Рис. 3.62 Аналіз роботи кодеру згорткового коду з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$ у разі надходження на його вхід кодової послідовності 1 0 1

Продовжимо аналіз послідовностей згорткового коду для наступних тактів роботи кодувального пристрою.

У момент часу $t = t_3$ надходить третій символ вхідного повідомлення

$m_3 = 1$, і це значення завантажується в перший біт вхідного регістру, а решта значень, як і на другому такті, зсується на 1 біт праворуч. Результат цієї операції буде наступним.

1. Вміст регістру – 101, що відповідає вхідній кодовій послідовності.

2. Результат на виході першого суматора – $u_1 = 1 \oplus 0 \oplus 1 = 0$.

3. Результат на виході другого суматора – $u_2 = 1 \oplus 1 = 0$.

4. Вихідна послідовність – 00.

У момент часу $t = t_4$ передавання вхідного повідомлення закінчується, тому ліві біти вхідного регістру тепер заповнюються нулями. Зокрема, безпосередньо для моменту часу $t = t_4$ маємо такий результат.

1. Вміст регістру – 010.

2. Результат на виході першого суматора – $u_1 = 0 \oplus 1 \oplus 0 = 1$.

3. Результат на виході другого суматора – $u_2 = 0 \oplus 0 = 0$.

4. Вихідна послідовність – 00.

Для моменту часу $t = t_5$, відповідно, маємо.

1. Вміст регістру – 001.

2. Результат на виході першого суматора – $u_1 = 0 \oplus 0 \oplus 1 = 1$.

3. Результат на виході другого суматора – $u_2 = 0 \oplus 1 = 1$.

4. Вихідна послідовність – 11.

І нарешті, для моменту часу $t = t_6$ можна записати.

1. Вміст регістру – 000.

2. Результат на виході першого суматора – $u_1 = 0 \oplus 0 \oplus 0 = 0$.

3. Результат на виході другого суматора – $u_2 = 0 \oplus 0 = 0$.

4. Вихідна послідовність – 00.

Для згорткового коду у разі, якщо вхідний регістр містить послідовність, яка складається лише з нулів, введення повідомлення можна вважати закінченим.

Результати роботи кодеру згорткового коду з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$ у разі надодження на його вхід кодової послідовності 1 0 1 для всіх моментів

часу, від $t = t_1$ до $t = t_6$, показані на рис. 3.61.

Остаточно, згортковий код з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$ для вхідної послідовності 1 0 1 складає 11 10 00 10 11.

Розглянемо приклад роботи цієї самої схеми кодеру для іншої вхідної послідовності із трьох бітів.

Приклад 3.53. Проаналізувати результати роботи кодеру, схема якого наведена на рис. 3.61, у різні моменти часу, вважаючи, що на його вхід подана послідовність бітів 1 1 0.

Будемо розв'язувати це завдання з урахуванням результатів, отриманих у прикладі 3.52. Стан вхідного регістру R та символи кодового слова u_1 та u_2 у відповідні моменти часу будуть мати наступні значення:

1. $t = t_1$, $m_1 = 1$, R = 1 0 0, $u_1 = 1 \oplus 0 \oplus 0 = 1$, $u_2 = 1 \oplus 0 = 1$.
2. $t = t_2$, $m_2 = 1$, R = 1 1 0, $u_1 = 1 \oplus 1 \oplus 0 = 0$, $u_2 = 1 \oplus 0 = 1$.
3. $t = t_3$, $m_3 = 0$, R = 0 1 1, $u_1 = 0 \oplus 1 \oplus 1 = 0$, $u_2 = 1 \oplus 0 = 1$.
4. $t = t_4$, $m_4 = 0$, R = 0 0 1, $u_1 = 0 \oplus 0 \oplus 1 = 1$, $u_2 = 1 \oplus 0 = 1$.
5. $t = t_5$, $m_4 = 0$, R = 0 0 0, $u_1 = 0 \oplus 0 \oplus 0 = 0$, $u_2 = 0 \oplus 0 = 0$.

Оскільки всі біти вхідного регістру R дорівнюють нулю, вже на п'ятому такті передавання повідомлення можна вважати завершеним. Тобто, шостий такт передавання повідомлення, який був необхідним у прикладі 3.52, у даному разі є зайвим.

Результати роботи кодеру завадостійкого коду з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$ для вхідної послідовності 1 0 1 наведені на рис. 3.63, а отримана кодова послідовність становить 11 01 01 11.

Результати, отримані у прикладах 3.52 та 3.53, показують, що за умови однакової довжини вхідної послідовності довжина вихідної послідовності згорткового коду, насправді, може бути різною.

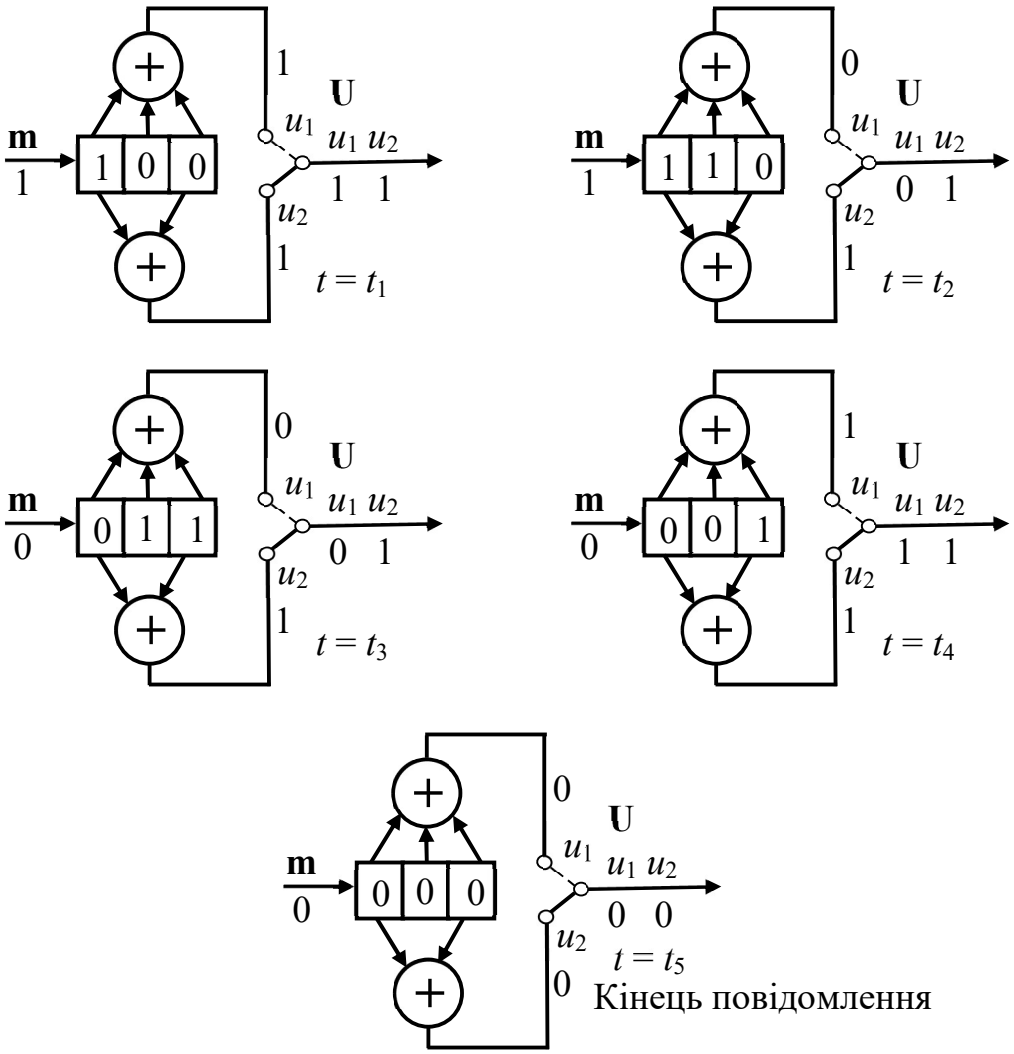


Рис. 3.63 Аналіз роботи кодеру згорткового коду з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$ у разі надходження на його вхід кодової послідовності 1 1 0

Приклад 3.54. Визначити коефіцієнти надлишковості згорткових кодів, розглянутих у прикладах 3.52 та 3.53, та порівняти ці значення із коефіцієнтом надлишковості лінійного коду із такими ж самими коректувальними параметрами.

Якщо розрахувати коефіцієнт надлишковості згорткового коду за формулою (2.38), для прикладів 3.52 та 3.53, відповідно, отримуємо:

$$\rho_1 = \frac{n-k}{n} = \frac{10-3}{10} = \frac{7}{10} > \frac{1}{2}; \qquad \rho_2 = \frac{n-k}{n} = \frac{8-3}{8} = \frac{5}{8} > \frac{1}{2}.$$

З іншого боку, для лінійного коду з двома символами кодового слова на

1 символ інформаційного повідомлення, як було використано для формування згорткового коду із параметром $\frac{1}{n} = \frac{1}{2}$, маємо інший результат:

$$\rho_л = \frac{n-k}{n} = \frac{2k-k}{2k} = \frac{k}{2k} = \frac{1}{2}.$$

Наприклад, у разі трьох інформаційних символів, як у прикладах 3.52 та 3.53, для лінійного коду: $k_л = 3$, $n_л = 6$, $\rho_л = \frac{1}{2}$. Взагалі згорткові коди завжди є більш надлишковими, ніж лінійні завадостійкі коди з відповідними коректувальними параметрами. З фізичної точки зору це пов'язано з тим, що після передавання всіх символів повідомлення необхідно ще провести через вхідний регістр відповідний ланцюжок із бітами отриманої вхідної інформації, тому кількість тактів збільшується від 3 до 6 у прикладі 3.52 та до 5 у прикладі 3.53. З іншого боку, якщо регістр заповнений нулями, передавання інформації можна вважати завершеним, тому довжина кодового слова згорткового коду може змінюватись залежно від вхідного повідомлення, яке кодується [33].

3.8.4 Головні способи подання згорткових кодів

3.8.4.1 Поліноміальне подання

У деяких випадках зв'язки між бітами вхідного регістру да суматорами за модулем два у згортковому коді зручно подавати у поліноміальній формі. Такі поліноми аналогічні поліноміальним генераторам циклічного коду, які були розглянуті у підрозділі 2.5. Згортковий код можна подати у вигляді набору з n поліноміальних генераторів, по одному генератору для кожного з n суматорів за модулем два. Кожний поліном має порядок $K - 1$ або менший та описує зв'язки кодувального регістру зсуву із відповідними суматорами за модулем два. Ці зв'язки відповідають векторам, зображеним на узагальненій структурній схемі декодера. Твірний поліном формується для згорткового коду наступним чином: якщо зв'язок між відповідним бітом регістра та суматором відсутній, коефіцієнт поліному для відповідної степені змінної X дорівнює 0, а якщо такий зв'язок наявний – цей коефіцієнт дорівнює 1.

Наприклад, для схеми, зображеної на рис. 3.61, можна записати два твірних поліноми, які відображують верхні та нижні зв'язки кодера, у наступній формі:

$$\mathbf{g}_1(X) = 1 + X + X^2; \qquad \mathbf{g}_2(X) = 1 + X^2.$$

Символи згорткового коду u_1 та u_2 формуються як результат множення елементів вектору вхідної послідовності **m** на твірні поліноми $\mathbf{g}_1(X)$ та $\mathbf{g}_2(X)$, для здійснення цієї операції вектор **m** також записується у поліноміальній формі.

Розглянемо приклад формування кодових послідовностей згорткового коду.

Приклад 3.55. Сформувати послідовності згорткового коду для кодера з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$, схема якого на ведена на рис. 3.61, та вхідних послідовностей **m** = 1 0 1 та **m** = 1 1 0.

Для вхідної послідовності **m** = 1 0 1 можна записати.

$\mathbf{m}(X) = 1 + X^2$;

$u_1(X) = \mathbf{m}(X)\mathbf{g}_1(X) = (1 + X^2)(1 + X + X^2) = 1 + X + X^2 + X^2 + X^3 + X^4 = 1 + X + X^3 + X^4.$

$u_2(X) = \mathbf{m}(X)\mathbf{g}_2(X) = (1 + X^2)(1 + X^2) = 1 + X^2 + X^2 + X^4 = 1 + X^4.$

$\mathbf{U}(X) = (1, 1) + (1, 0)X + (0, 0)X^2 + (1, 0)X^3 + (1, 1)X^4.$

U = 11 10 00 10 11.

Відповідно, для вхідної послідовності **m** = 1 1 0 маємо.

$\mathbf{m}(X) = X + X^2$;

$u_1(X) = \mathbf{m}(X)\mathbf{g}_1(X) = (X + X^2)(1 + X + X^2) = X + X^2 + X^3 + X^2 + X^3 + X^4 = X + X^4.$

$u_2(X) = \mathbf{m}(X)\mathbf{g}_2(X) = (X + X^2)(1 + X^2) = X + X^2 + X^3 + X^4.$

$\mathbf{U}(X) = (1, 1)X + (0, 1)X^2 + (0, 1)X^3 + (1, 1)X^4.$

U = 11 01 01 11.

Слід відзначити, що біти вихідної послідовності у разі поліноміального подання циклічного коду записані у зворотному порядку.

3.8.4.2 Подання кодера згорткового коду у вигляді

діаграми станів скінченного автомату

Для описання принципу роботи згорткового коду може бути використана теорія скінченних автоматів, яка була розглянута у третьому томі другої частини посібника [50]. Нагадаємо, що сутність цієї теорії полягає у тому, що цифровий або програмований електронний пристрій описується відповідним станом із заданого скінченного набору. Тоді стан скінченного автомату у наступний момент визначається двома факторами: вхідною дією та набором попередніх станів із кількістю k, де k – кількість станів у пам'яті автомату.

Для описання структури скінченного автомату із заданою кількістю станів використовують діаграми станів [50]. Розглянемо діаграму станів для кодера згорткового коду з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$, структурна схема якого на ведена на рис. 3.61. Стани, які необхідно відобразити та проаналізувати у рамках цієї діаграми, являють собою вміст граничних K – 1 правих бітів вхідного регістра, що відповідає двом бітам. Відповідно, у даному випадку буде 4 стани автомату із значеннями: a = 00, b = 10, c = 01, d = 11. Тоді структурну схему кодеру, наведену на рис. 3.61, можна представити у вигляді діаграми станів скінченного автомату, яка показана на рис. 3.64 [33].

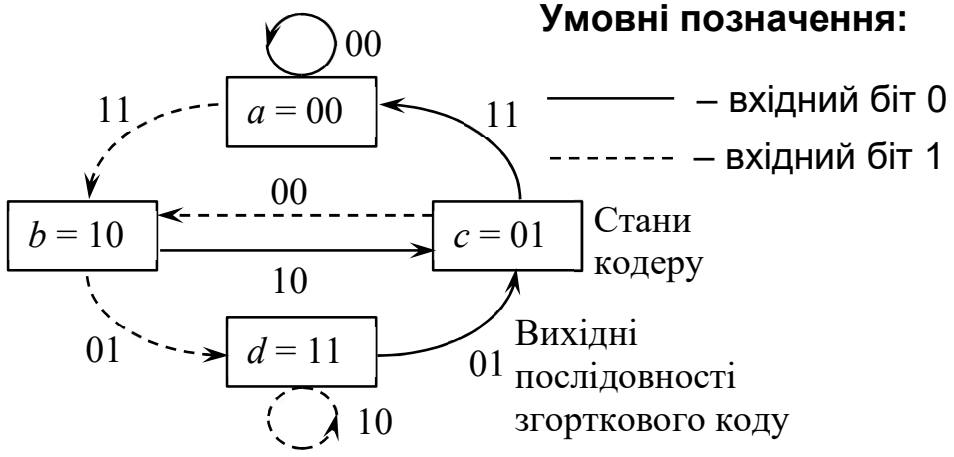


Рис. 3.64 Діаграма станів кодеру, схема якого наведена на рис. 3.61

Крім діаграми станів, яка відображує можливі переходи між

визначеними станами скінченного автомату, для аналізу станів автомату у фіксовані моменти часу використовують табличне подання [50]. Для заповнення таких таблиць фіксується початковий, або нульовий стан автомату, та задається вхідна послідовність цифрових сигналів. Наприклад, початковим станом автомату може бути вимкнений стан [50], тоді серед набору подій має бути така, яка здатна вивести автомат із цього стану. Наприклад, це може бути кнопка вмикання або надходження відповідного сигналу [50]. Для діаграм станів кодера згорткового коду, аналогічних наведеній на рис. 3.64, початковим станом зазвичай вважається стан, у якому всі біти вхідного регістру дорівнюють 0. Якщо відомий початковий стан автомату, його стани у наступні моменти часу визначаються за діаграмою станів через відому послідовність вхідних сигналів.

Розглянемо відповідні приклади.

Приклад 3.56. З використанням діаграми станів, наведеної на рис. 3.64, сформувати згортковий код для вхідної послідовності **m** = 11011, вважаючи що у початковому стані всі біти вхідного регістру дорівнюють 0.

Результати розв’язування цієї задачі, згідно із схемою скінченного автомату, наведеною на рис. 3.64, представлені у таблиці 3.25.

Зрозуміло, що стани скінченного автомату залежать не лише від вхідної послідовності бітів, але й від стану регістру R. Розглянемо наступний приклад.

Приклад 3.57. З використанням діаграми станів, наведеної на рис. 3.64, сформувати згортковий код для вхідної послідовності **m** = 11011, вважаючи що у початковому стані два старші біти вхідного регістру дорівнюють 1, а значення молодшого біту є невизначеним.

Результати розв’язування цієї задачі згідно із схемою скінченного автомату, наведеною на рис. 3.64, представлені у таблиці 3.26.

Таблиця 3.25 – Результати роботи схеми скінченного автомату,

наведеної на рис. 3.64, за умови вхідної кодової послідовності **m** = 11011 та початкового значення 0 для всіх бітів вхідного регістру

Вхідні біти, m_i	Стан регістру R	Стан схеми у момент часу t_i	Стан схеми у момент часу t_{i+1}	Кодове слово у момент часу t_i	
				u_1	u_2
–	0 0 0	0 0	0 0	–	–
1	1 0 0	0 0	1 0	1	1
1	1 1 0	1 0	1 1	0	1
0	0 1 1	1 1	0 1	0	1
1	1 0 1	0 1	1 0	0	0
1	1 1 0	1 0	1 1	0	1
0	0 1 1	1 1	0 1	0	1
0	0 0 1	0 1	0 0	1	1
Вихідна послідовність бітів: 11 01 01 00 01 01 11					

Таблиця 3.26 – Результати роботи схеми скінченного автомату, наведеної на рис. 3.64, за умови вхідної кодової послідовності **m** = 11011 та початкового значення 1 для перших двох бітів вхідного регістру

Вхідні біти, m_i	Стан регістру R	Стан схеми у момент часу t_i	Стан схеми у момент часу t_{i+1}	Кодове слово у момент часу t_i	
				u_1	u_2
–	1 1 –	1 –	1 1	–	–
1	1 1 1	1 1	1 1	1	0
1	1 1 1	1 1	1 1	1	0
0	0 1 1	1 1	0 1	0	1
1	1 0 1	0 1	1 0	0	0
1	1 1 0	1 0	1 1	0	1
0	0 1 1	1 1	0 1	0	1
0	0 0 1	0 1	0 0	1	1
Вихідна послідовність бітів: 10 10 01 00 01 01 11					

3.8.4.3 Деревоподібні діаграми

Зрозуміло, що діаграма станів скінченного автомату, аналогічна наведеній на рис. 3.64, за умови відомих значень бітів вхідного регістру R повністю описує роботу схеми формування згорткового коду у будь-який момент часу. Проте, як було відмічено у третьому томі другої частини посібника, суттєвим недоліком схеми скінченного автомату є те, що вона не відображує зміни станів автомату за умови надходження на її вхід відповідної цифрової кодової послідовності [50]. Тому для аналізу зміни станів скінченного автомату у часі частіше використовують не діаграми станів, а створені на основі таких діаграм деревоподібні діаграми (англійський термін – tree diagram), на яких можна спостерігати часову зміну станів автомату [50]. Саме такий підхід використовується і для аналізу особливостей роботи згорткових кодів [33]. Деревоподібна діаграма для схеми скінченного автомату, наведеної на рис. 3.64, відображена на рис. 3.65.

Структуру деревоподібної діаграми, наведеної на рис. 3.65, необхідно аналізувати, використовуючи наступне правило. Якщо прийнятий вхідний біт дорівнює 0, перехід здійснюється за верхньою гілкою, а якщо цей біт дорівнює 1 – за нижньою гілкою. Відповідні напрямки руху вздовж гілок показані на першій гілці, розташованій у корені дерева. Головним препущеням для формування деревоподібної діаграми згорткового коду є те, що на початку роботи скінченного автомату у вхідному регістрі R міститься нульове значення. Тоді, згідно із діаграмою станів скінченного автомату, яка наведена на рис. 3.64, якщо в першому такті на вхід кодеру надходить символ 0, на виході формується кодова послідовність 00, а якщо у цей момент часу на вхід надходить символ 1, на виході формується послідовність 11. Загальне правило аналізу деревоподібної діаграми полягає у тому, що, якщо на вхід автомату надходить символ 0, переміщення за гілкою дерева здійснюється вниз, а якщо символ 1 – вгору. На рис. 3.65 показаний аналіз стану кодеру з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$ за умови надходження послідовності символів 1 1 0 1 1.

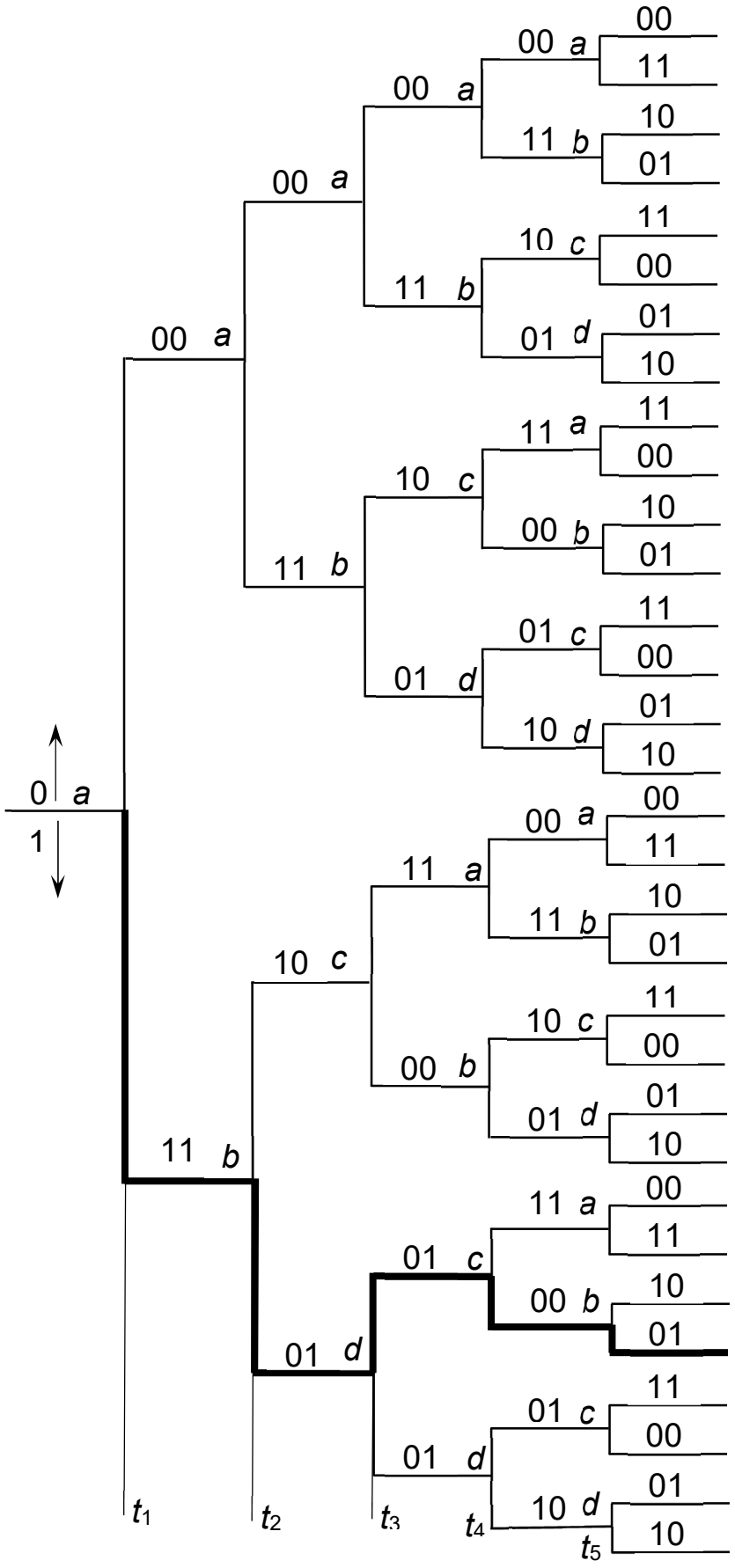


Рис. 3.65 Деревоподібна діаграма для скінченного автомату, діаграма станів якого показана на рис. 3.64

Згідно із структурою деревоподібної діаграми, наведеною на рис. 3.65, нулю на вході відповідає символ *a*, а одиниці – символ *b*. Особливості використання пам'яті скінченного автомату полягають у тому, що бітові послідовності із двох символів також мають відповідні двійкові коди. Послідовності вхідних бітів 01 відповідає символ *c* із кодом 01, а вхідній послідовності 11 – символ *d* із кодом 11.

Зрозумівши описаний вище принцип побудови деревоподібної діаграми, можна знайти на ній шлях, який відповідає вхідний кодовій послідовності 1 1 0 1 1. Від повідний шлях показаний на рис. 3.65 жирними лініями. Дійсно, на обраному шляху одиницям вхідної кодової послідовності відповідає переміщення вниз за гілками кодового дерева, а нулю – переміщення догори. Тобто, загальне правило формування та аналізу деревподібної діаграми виконане.

3.8.4.4 Ґраткові діаграми

Аналіз деревоподібної діаграми згорткового коду, наведеної на рис. 3.65, показує, що структура дерева є циклічною та повторюється із періодом, який відповідає бітовому розміру *K* вхідного регістру R. Як видно з рис. 3.65, перше розгалуження дерева дає лише два символи, *a* та *b*, але на другому розгалуженні, у момент часу t_2, до них додається ще два символи, *c* та *d*. Під час третього розгалуження, якому відповідає момент часу t_3, формується вже вісім вузлів: два вузла *a*, два вузла *b*, два *c* та два *d*. На четвертому розгалуженні у момент часу t_3 утворюється 16 вузлів: чотири вузла *a*, чотири вузла *b*, чотири *c* та чотири *d*. Тобто, після кожного розгалуження кількість вузлів дерева збільшується вдвічі, незважаючи на те, що кількість станів системи є постійною і дорівнює 4. Тобто, з урахуванням значення $K = 3$, на третьому розгалуженні стани системи починають повторюватись.

Тому в теорії кодування існує також інший підхід до описання структури

згорткового коду, пов’язаний із використанням ґраткових діаграм. Загальні основи теорії ґраток були розглянуті у першому томі другої частини посібника [48].

Розглянемо стани кодеру, діаграма станів якого наведена на рис. 3.64, а деревоподібна діаграма – на рис. 3.65, у початковий момент часу t_1. Зрозуміло, що можливими є два вихідних стана кодеру: стан 00, у випадку, коли перший вхідний біт дорівнює 1, та 11, у випадку, коли перший вхідний біт дорівнює 0. Проте вже у момент часу t_2 ситуація дещо ускладняється. Тут можливими є чотири стани вхідного регістру: 0 0 0, 1 0 0, 0 1 0 та 1 1 0. Побудуємо на основі цих даних ґраткову діаграму кодеру. Припустимо, що нульовому біту відповідає неперервна лінія, а одиничному біту – пунктирна. Тоді узагальнену ґраткову діаграму кодеру, діаграма станів якого наведена рис. 3.64, можна подати у вигляді, графу, який наведений на рис. 3.66.

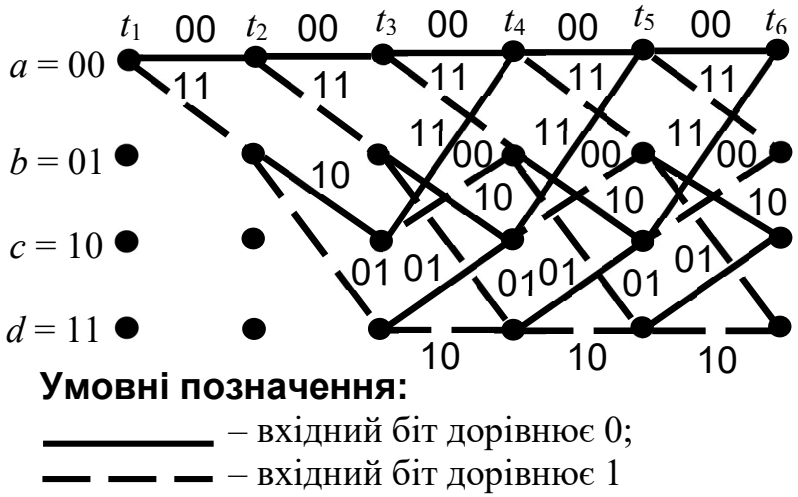


Рис. 3.66 Приклад ґраткової діаграми для згорткового коду, який генерується схемою скінченного автомату, що наведена на рис. 3.64

Проведемо аналіз узагальненого способу формування ґраткової діаграми згорткового коду, аналогічної представленій на рис. 3.66. Як було показано вище, під час побудови ґраткової діаграми використані ті самі умовні позначення, що й для діаграми станів скінченного автомату. Суцільною лінією позначені вхідні дані, які відповідають значенню вхідного біту 0, а пунктирною – дані, які відповідають значенню вхідного біту 1. Вузли ґратки

відображують стани кодеру. Перший рядок вузлів відповідає стану $a = 00$, другий – стану $b = 01$, третій – стану $c = 10$, а останній – стану $d = 11$. У кожний момент часу для відображення 2^{K-1} станів системи необхідно така сама кількість вузлів ґратки. Проте, як видно з рис. 3.66, за умови $n = K - 1$ ситуація ретельно змінюється і кількість вузлів вже не збільшується. Навпаки, ґратка починає мати чітко зафіксовану періодичну структуру. Починаючи з цього моменту можна із кожного стану скінченного автомату перейти до двох з попередніх станів, і цей перехід визначається вхідним сигналом. І навпаки, кожний із станів автомату може бути досягнутий через переходи з двох попередніх, відповідні переходи залежать від наявності сигналів 0 або 1 на вході системи. Тобто, один стовпець часового інтервалу після моменту часу $t = t_3$ цілком визначає згортковий код у наступні моменти часу. Проте, на ґратковій діаграмі, наведеній на рис. 3.66, показані гілки не лише для моменту часу $t = t_3$, але й для моментів $t = t_5$ та $t = t_6$, хоча зрозуміло, що діаграма вже є періодичною, а відповідні гілки – паралельними. Чи варто в такому разі відображати всю структуру ґратки, якщо вона повністю визначається своїми початковими гілками, до моменту часу $t = t_3$?

Тут в теорії згорткового кодування не існує єдиної думки. Деякі автори вважають, що за такої умови дійсно достатньо передати стани автомату лише за умови $t \le t_K$, тоді для значень $t > t_K$ їх можна відтворити автоматично. Але, з іншого боку, ґраткові діаграми згорткових кодів, створені також для моментів часу $t > t_K$, простіше використовувати для визначення остаточного стану кодеру за умови великих значень t [33, 86].

Розглянуті у цьому підрозділі ґраткові діаграми мають суттєве значення для розуміння способу роботи декодерів згроткових кодів, які працюють за принципом максимуму правдоподібності [49]. Відповідний теоретичний матеріал пов’язаний із визначенням метріки кодової послідовності за Хеммінгом та буде розглянутий у наступному підрозділі. Поняття про метрики кодових послідовностей є одним із базових понять векторного аналізу, воно було надане у першому томі другої частини посібника [48].

3.8.5 Метрики шляхів згорткового коду та пошук помилок за метриками

Повернемося до рис. 3.66, на якому представлено ґраткова діаграма згорткового коду. Проаналізуємо ґраткову діаграму, яка наведена на рис. 3.66, та визначимо відстані між прийнятою кодовою комбінацією та очікуваною комбінацією згідно із структурою ґраткової діаграми. Будемо вважати, що відстані між кодовими комбінаціями відповідають метриці за Хеммінгом. Відповідне поняття є основою векторного аналізу лінійних кодів та було надано у п’ятому розділі першого тому другої частини посібника [48]. Згідно із аналізом кодового дерева будемо формувати нову ґраткову діаграму, на кожній гілці якої позначимо відстань за Хеммінгом між прийнятою послідовністю згорткового коду та послідовністю, яка очікується відповідно до структури кодового дерева. Зробити це буде не важко, оскільки очікувані послідовності вже були позначені на рис 3.66.

Розглянемо спочатку роботу декодеру згорткового коду у разі відсутності помилок. Відповідна ґраткова діаграма для вхідної послідовності **m** = 1 1 0 1 1 із позначеними метриками Хеммінга для кожного моменту часу роботи декодеру показана на рис. 3.67. Проаналізуємо її.

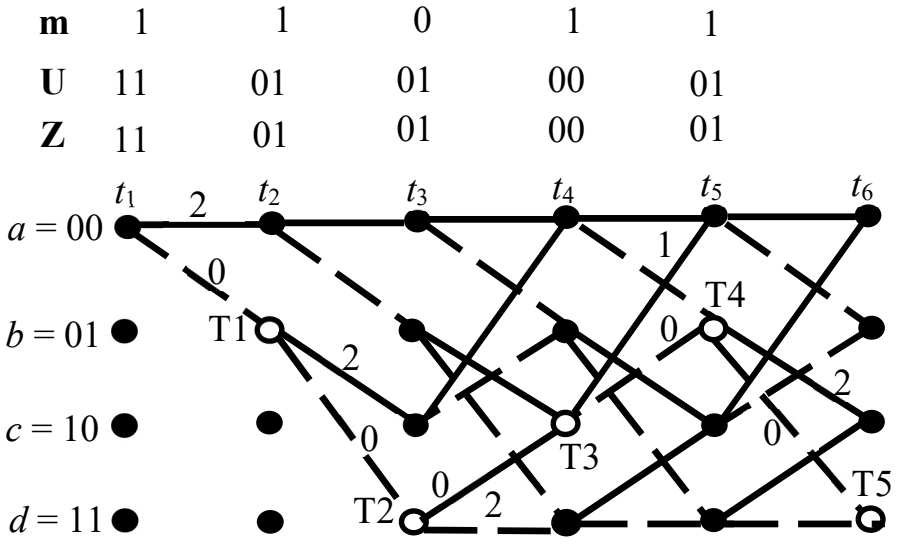


Рис. 3.67 Ґраткова діаграма декодеру згорткового коду із визначенням відстані за Хеммінгом між вихідними та очікуваними кодовими комбінаціями

Відповідно до алгоритму роботи кодеру, першим кодовим словом буде 00. Після надходження нового символу кожна гілка кодової ґратки помічається позначкою подібності. Кодова відстань між послідовністю символів 00, яка була до моменту часу t_1, і послідовністю 11, що надійшла, складає 2. У зв'язку з цим відрізок верхньої гілки граткової діаграми між точками t_1 та t_2 помітимо цифрою 2. Розглянемо тепер нижню гілку. Тут стан 01 точно відповідає прийнятому кодовому слову 11, відповідно, кодова відстань для цієї гілки дорівнює 0. Тобто, аналіз ґраткової діаграми, наведеної на рис. 3.67, показує, що у момент часу t_1 мінімальна кодова відстань за Хеммінгом між вхідною та очікуваною послідовністю дорівнює 0. Ця відстань відповідає точці Т1, яка на ґратковій діаграмі помічена білим кольором.

Розглянемо тепер наступний момент часу t_3. Згідно із рис. 3.66, із точки Т1 виходить дві гілки. Першій з них відповідає послідовність 10, а другій послідовність 01. Оскільки прийнята послідовність 01, для першої з цих гілок кодова відстань між прийнятою та очікуваною комбінаціями складає 2, а для другої – 0. Відповідно, можна вважати, що правильний шлях веде через другу розглянуту гілку дерева, тоді наступним проміжним вузлом є точка Т3. Відповідні кодові відстані для обох гілок ґраткової діаграми також вказані на рис. 3.67.

Стан декодеру в момент часу t_4 визначимо наступним чином. Із точки Т3 також виходять дві гілки. Згідно з рис. 3.66, перша з них відповідає значенню 0 на вході кодеру та вихідній кодовій послідовності 01, а друга – значенню 1 на вході кодеру та вихідній кодовій послідовності 11. Відповідно, для першої гілки відстань між прийнятим та очікуваним кодовими словами складає 0, а для другої це значення дорівнює 1. Тому наступним проміжним вузлом на ґратковій діаграмі, наведеній на рис. 3.67, буде точка Т4, оскільки їй відповідає менша кодова відстань. І в решті решт, проаналізуємо стан декодеру та прийнятий вхідний сигнал у момент часу t_5. З точки Т4 також виходять дві гілки, перша з яких, згідно з рис. 3.66, відповідає значенню вхідного символу

0 та кодовій послідовності 10, а друга – значенню вхідного символу 1 та кодовій послідовності 01. Порівнюючи ці значення із значенням прийнятої послідовності символів 01, визначаємо, що для першої гілки кодова відстань між прийнятою та очікуваною кодовою комбінацією дорівнює 2, а для другої цей параметр становить 0. Відповідно, згідно з правилом мінімальної кодової відстані, обираємо другу гілку дерева, яка веде до вузла T5. На цьому аналіз дерева закінчений. Оскільки для всіх обраних гілок ґраткової діаграми відстань Хеммінга дорівнює 0, визначений шлях та прийняту кодову комбінацію **U** = 11 01 01 00 01 можна вважати правильними.

Сумарна кодова відстань всіх гілок для шляху, визначеного на кодовому дереві, називається сумарною метрикою шляху за Хеммінгом. Якщо кодова послідовність є правильною, сумарна метрика її шляху завжди дорівнює 0.

Розглянемо тепер інший випадок. Припустимо, що замість послідовності $\mathbf{U}_{cor}$ = 11 01 01 00 01 прийнята спотворена послідовність $\mathbf{U}_{err}$ = 11 01 11 00 01, тобто, у третій парі прийнятих бітів виникла помилка. Відповідна ґраткова діаграма показана на рис. 3.68.

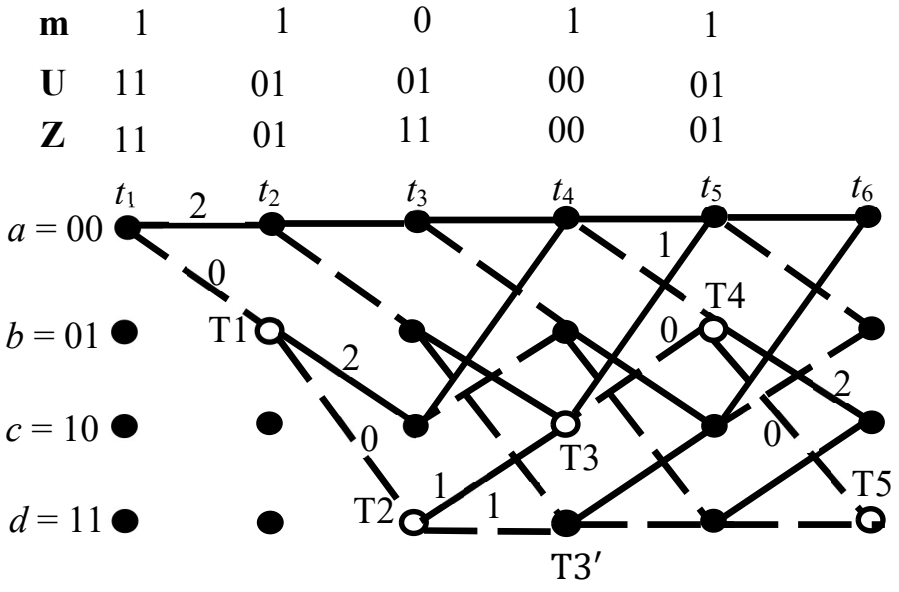


Рис. 3.68 Ґраткова діаграма декодеру згорткового коду за умови помилки у третій парі прийнятих бітів переданої кодової послідовності

Зрозуміло, що до вузлової точки T2 аналіз ґраткової діаграми не буде відрізнятися від попереднього. Але у точці T2 виникає проблема вибору. Річ у

тому, що прийнята кодова послідовність 11 є неправильною, оскільки вона не відповідає діаграмі станів скінченного автомату, наведеній на рис. 3.64. Для стану автомату, визначеного вузлом Т2, можливими є або кодова послідовність 01, або кодова послідовність 11, відповідні переходи між станами показані на ґратковій діаграмі кодеру згорткового коду, наведеній на рис. 3.66. Проте проблемність ситуації полягає у тому, що у випадку, який розглядається, кодова відстань для обох гілок дерева складає 1 і не можна визначити, який з двох можливих шляхів є правильним. Для розв'язування цієї проблеми пошуку помилки у теорії згорткових кодів використовується метод аналізу структури дерева із поверненням до попередніх вузлових точок. Розглянемо, як працює цей метод на прикладі декодера із визначеною структурою та заданої спотвореної кодової послідовності. Припустимо, що із двох розгалужених гілок, що йдуть від точки Т2, вага яких є однаковою та дорівнює 1, обрана хибна гілка, яке веде до вузлової точки Т3′. Це означає, що в момент часу t_3 замість правильного закодованого символу 0 прийняте хибне значення 1. Розглянемо за цієї умови роботу декодеру у наступний момент часу t_4. Як видно з граткової діаграми, яка наведена на рис. 3.64, вузловій точці Т3′ відповідають вхідні сигнали 01 та 10, і кодова відстань між цими кодовими словами та прийнятою послідовністю 00 складає 1. А якщо повернутися до аналізу точки Т3 та відповідних метрик, тоді вага гілки, яка веде до точки Т3, є мінімальною та складає 0. Тобто, у разі виникнення помилкової ситуації з метою уникнення ефекту накопичення помилок слід аналізувати всю структуру дерева з самого початку.

Приклад 3.58. Побудувати діаграму станів та ґраткову діаграму для схеми формування згорткового коду з параметрами $K = 5$ та $\frac{1}{n} = \frac{1}{3}$, наведеної на рис. 3.69.

Почнемо розв'язування цієї задачі із розгляду можливих станів кодувальної схеми, наведеної на рис. 3.69. Зважаючи на те, що довжина вхідного регістру становить $K = 5$ бітів, кількість станів для такої схеми визначається як $2^{K-1} = 2^4 = 16$. Позначмо ці можливі стани малими латинськими

літерами від *a* до *p* наступним чином.

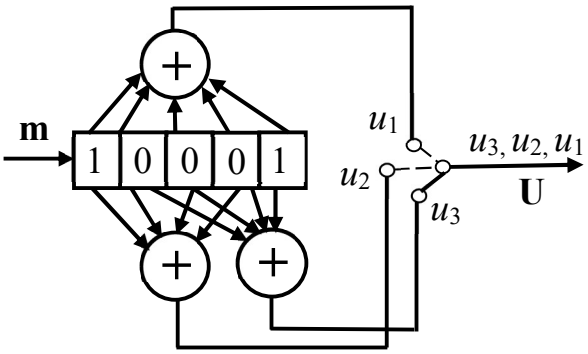


Рис. 3.69 Структурна схема кодеру згорткового коду з параметрами $K = 5$ та $\frac{1}{n} = \frac{1}{3}$ для прикладу 3.58

a = 0000; *b* = 1000; *c*= 0100; *d* = 1100; *e* = 0010; *f* = 1010; *g* = 0110; *h* = 1110; *i* = 0001; *j* = 1001; *k* = 0101; *l* = 1101; *m* = 0011; *n* = 1011; *o* = 0111; *p* = 1111.

Зверніть особливу увагу на те, що у згортковому коді порядок запису бітів двійкового числа є не прямим, а зворотним, оскільки саме так, зліва-направо, надходять біти інформаційного повідомлення до вхідного регістру. На відміну від цього, вихідна послідовність **U** для схеми, наведеної на рис.3.69, записана у прямому порядку. Як і для всіх інших типів кодів, що розглядалися раніше, порядок запису бітів числа у згортковому коді також має важливе значення для подальшого коректного розшифрування кодової комбінації.

Тепер, з урахуванням особливостей роботи схеми, наведеної на рис. 3.69, проаналізуємо можливі переходи між визначеними станами кодеру за умови значень вхідного сигналу *m* = 0 та *m* = 1. Можливі відповідні переходи між станами кодеру описані у таблиці 3.27.

Таблиця 3.27 – Можливі переходи між станами схеми формування згорткового коду, наведеної на рис. 3.69

Поточний стан	Вхідний сигнал	Стан, до якого переходить кодер	Вихідний сигнал
1. *a* = 0000	*m* = 0	*a* → *a* = 0000	**U** = 000
	m = 1	*a* → *b* = 1000	**U** = 011

Таблиця 3.27 (продовження)

Поточний стан	Вхідний сигнал	Стан, до якого переходить кодер	Вихідний сигнал
2. $b = 1000$	$m = 0$	$b \rightarrow c = 0100$	**U** = 111
	$m = 1$	$b \rightarrow d = 1100$	**U** = 100
3. $c = 0100$	$m = 0$	$c \rightarrow e = 0100$	**U** = 111
	$m = 1$	$c \rightarrow f = 1010$	**U** = 100
4. $d = 1100$	$m = 0$	$d \rightarrow g = 0110$	**U** = 000
	$m = 1$	$d \rightarrow h = 1110$	**U** = 011
5. $e = 0010$	$m = 0$	$e \rightarrow i = 0001$	**U** = 111
	$m = 1$	$e \rightarrow j = 1001$	**U** = 100
6. $f = 1010$	$m = 0$	$f \rightarrow k = 0101$	**U** = 000
	$m = 1$	$f \rightarrow l = 1101$	**U** = 011
7. $g = 0110$	$m = 0$	$g \rightarrow m = 0011$	**U** = 000
	$m = 1$	$g \rightarrow n = 1011$	**U** = 011
8. $h = 1110$	$m = 0$	$h \rightarrow o = 0111$	**U** = 111
	$m = 1$	$h \rightarrow p = 1111$	**U** = 011
9. $i = 0001$	$m = 0$	$i \rightarrow a = 0000$	**U** = 101
	$m = 1$	$i \rightarrow b = 1000$	**U** = 110
10. $j = 1001$	$m = 0$	$j \rightarrow c = 0100$	**U** = 010
	$m = 1$	$j \rightarrow d = 1100$	**U** = 001
11. $k = 0101$	$m = 0$	$k \rightarrow e = 0010$	**U** = 000
	$m = 1$	$k \rightarrow f = 1010$	**U** = 001
12. $l = 1101$	$m = 0$	$l \rightarrow g = 0110$	**U** = 101
	$m = 1$	$l \rightarrow h = 1110$	**U** = 110
13. $m = 0011$	$m = 0$	$m \rightarrow i = 0001$	**U** = 010
	$m = 1$	$m \rightarrow j = 1001$	**U** = 001
14. $n = 1011$	$m = 0$	$n \rightarrow k = 0101$	**U** = 101
	$m = 1$	$n \rightarrow l = 1101$	**U** = 110

Таблиця 3.27 (закінчення)

Поточний стан	Вхідний сигнал	Стан, до якого переходить кодер	Вихідний сигнал
15. $o = 0111$	$m = 0$	$o \rightarrow m = 0011$	**U** = 101
	$m = 1$	$o \rightarrow n = 1011$	**U** = 110
16. $p = 1111$	$m = 0$	$p \rightarrow o = 0111$	**U** = 010
	$m = 1$	$p \rightarrow p = 1111$	**U** = 001

Переходи між станами кодеру, наведені у таблиці 3.27, можна відобразити у вигляді схеми скінченного автомату, яка наведена на рис. 3.70.

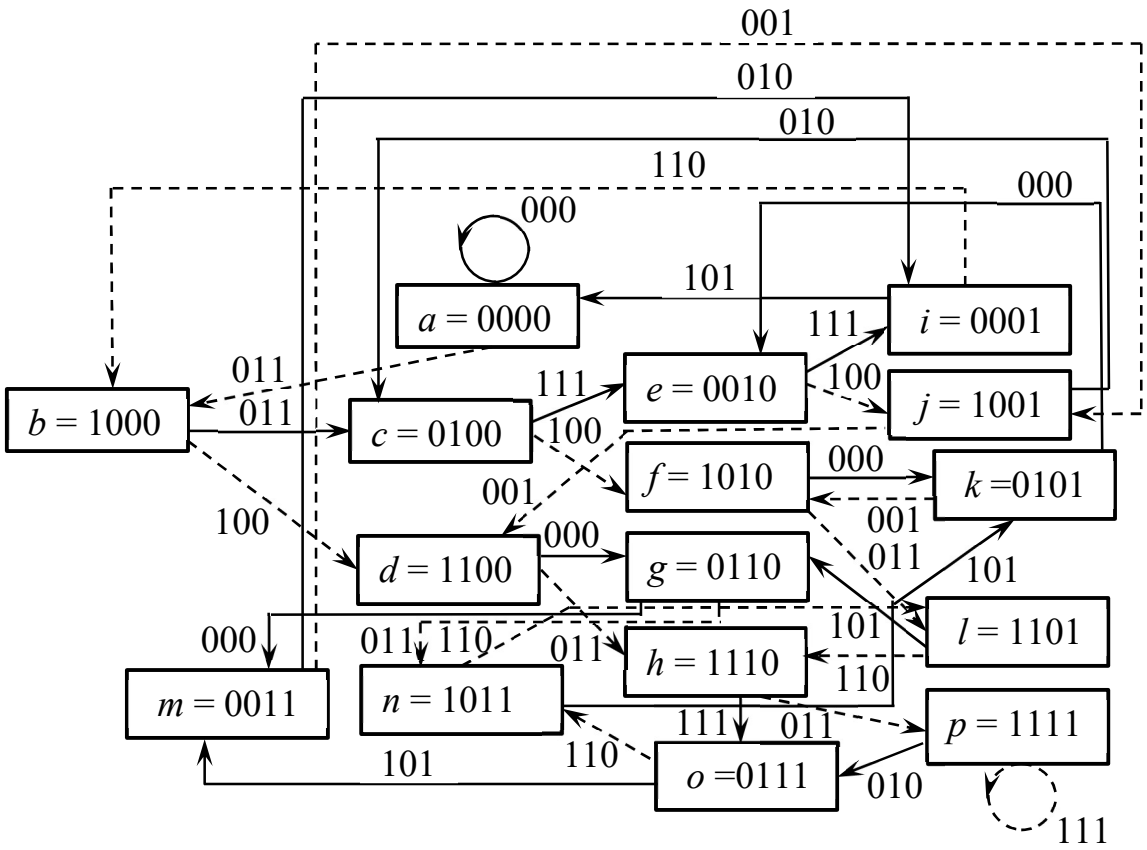


Рис. 3.70 Діаграма станів скінченного автомату для прикладу 3.58

Ґраткова діаграма схеми формування згорткового коду, діаграма станів якої у вигляді алгоритму роботи скінченного автомату показана на рис.3.70, наведена на рис. 3.71.

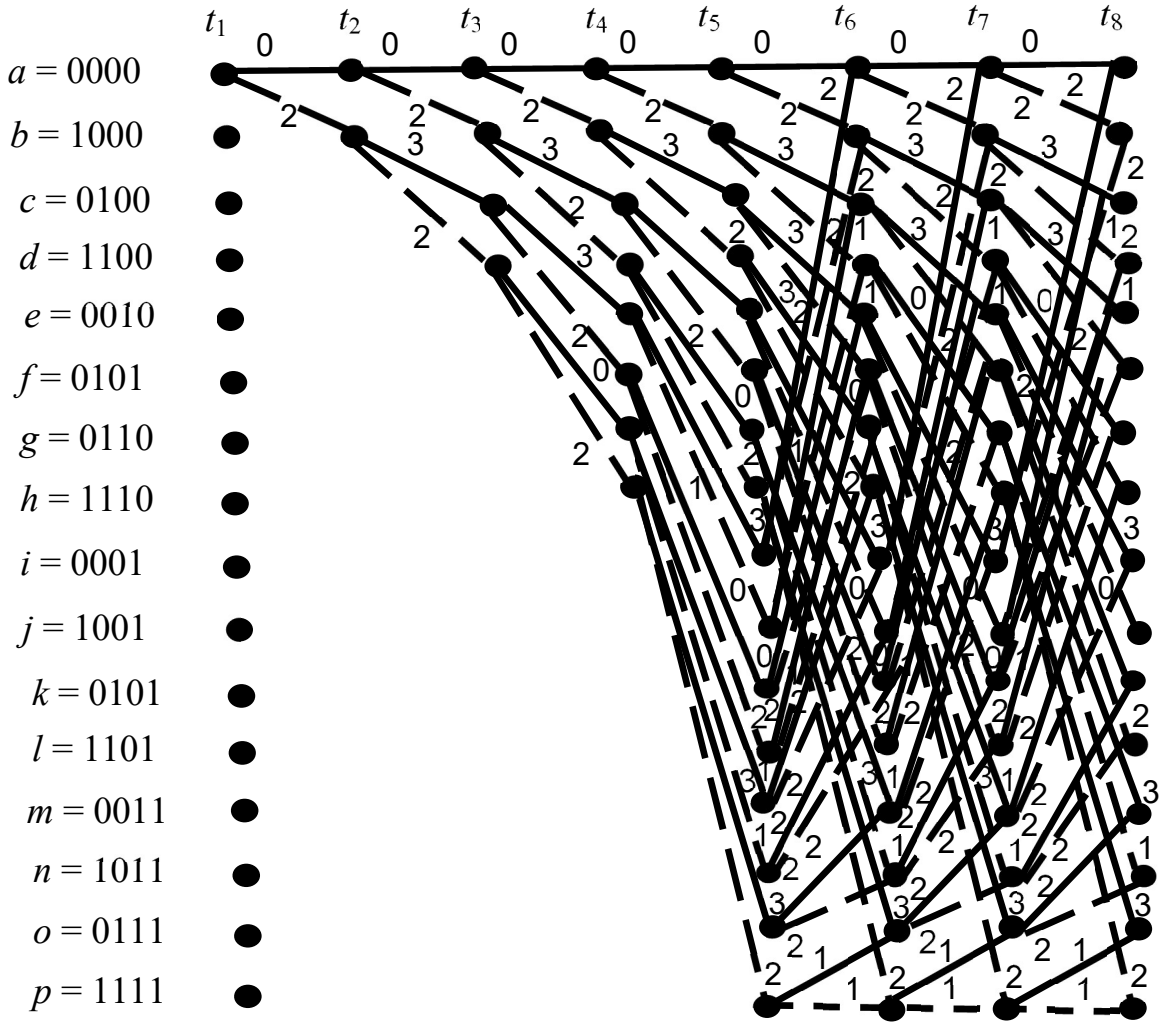


Рис. 3.71 Ґраткова діаграма схеми формування згорткового коду для прикладу 3.58

Розглянемо інший приклад, який оснований на результатах, отриманих у прикладі 3.58.

Приклад 3.59. Знайти кодову послідовність, яку формує кодер згорткового коду, розглянутий у прикладі 3.58, за умови надходження на його вхід послідовності бітів 10101. Вважати, що початковий стан системи становить 00000.

Будемо розв’язувати поставлену задачу, використовуючи числові дані, наведені у таблиці 3.7 та на ґратковій діаграмі, наведеній на рис. 3.71.

Якщо початковий стан системи становить *a*, у разі надходження на вхід символу 1 система переходить до стану *b* та формує перше кодове слово 011.

Тепер, у разі надходження сигналу 0 система формування згорткового коду переходить із стану *b* до стану *c* та формує вихідну послідовність 111. Далі, згідно з таблицею 3.7, зміна станів системи здійснюється наступним чином.

$t = t_3$, $\mathbf{m} = 1$, $c \to f$, $\mathbf{U} = 100$;
$t = t_4$, $\mathbf{m} = 0$, $f \to k$, $\mathbf{U} = 000$;
$t = t_4$, $\mathbf{m} = 1$, $k \to f$, $\mathbf{U} = 001$.

Тобто, кодова послідовність, яка формується кодером згорткового коду, схема якого наведена на рис. 3.69, за умови дії вхідного сигналу 10101 складає 011 111 100 000 001.

На основі прикладів 3.58 та 3.59 розглянемо особливості роботи декодеру згорткового коду у разі надходження на його вхід правильної та спотвореної кодової послідовності.

Приклад 3.60. Знайти кодову послідовність, яку буде сформовано на виході декодеру згорткового коду за умови надходження на його вхід послідовностей символів 011 111 100 000 001 та 011 101 100 000 001.

Ситуація, яка виникає у разі надходження на вхід декодеру послідовності символів 011 111 100 000 001 , була розглянута у прикладі 3.59. Ця вхідна послідовність є правильною та відповідає послідовності вхідних бітів кодеру 10101.

У випадку надходження на вхід декодеру спотвореної кодової послідовності 011 101 100 000 001 на другому такті роботи він знаходиться у стані *b* та приймає хибну кодову послідовність 101. Згідно з ґратковою діаграмою, наведеною на рис. 3.71, тут можливі два правильних варіанти: перехід достану *c* за умови $\mathbf{m} = 0$, або перехід до стану *d* за умови $\mathbf{m} = 1$. Умові $\mathbf{m} = 0$ відповідає кодова послідовність 111, вага якої складає 3, а умові $\mathbf{m} = 1$ – послідовність 100 та її вага становить 1. З урахуванням того, що вага прийнятої кодової комбінації складає 2, на цьому етапі декодування обрати правильний варіант неможливо. Припустимо, що була передана комбінація 100 та система перейшла до стану *d*. Тоді наступний код із трьох символів, 100, також є хибним, оскільки стану *d* відповідають інші кодові комбінації, а саме, 000 за

умови **m** = 0 та 011 за умови **m** = 1. Навпаки, якщо вважати, що на другому такті була передана комбінація 111 та система повинна перейти до стану *c*, подальші кодові комбінації розшифровуються безпомилково, згідно з прикладом 3.59. Тобто, можна вважати, що спотвореній на другому такті комбінації символів 101 відповідає послідовність 111.

Із розглянутих прикладів зрозуміло, що сутність алгоритмів декодування згорткового коду та пошуку помилок за структурою ґраткової діаграми полягає у тому, що сформований код є надлишковим і не всі кодові послідовності дозволені для визначених станів скінченного автомату. Тобто, для згорткових кодів, як і для лінійних, головним принципом формування є використання надлишкових кодів для створення відповідної кількості дозволених та заборонених кодових комбінацій. Відповідні теоретичні відомості щодо загальних принципів завадостійкого кодування були розглянуті у підрозділі 2.1. Відмінність принципу формування згорткових кодів полягає лише в тому, що дозволені та заборонені кодові комбінації декодеру визначаються принципом роботи та послідовністю станів скінченного автомату, який описує алгоритм роботи кодеру.

Проте пошук правильної кодової послідовності згорткового коду в спотвореній кодовій комбінації є досить складним завданням теорії імовірностей, теорії оптимізації та дискретної математики [25, 54, 55]. Відповідні алгоритми декодування будуть розглянуті у наступних підрозділах.

3.8.6 Алгоритми декодування згорткових кодів

3.8.6.1 Послідовне декодування

Вперше згорткові коди та можливості їхнього використання у системах зв’язку були розглянуті у роботах радянського математика та радіотехніка Л.М. Фінка та американського математика П. Елайєса [33, 69, 86].

Лев Матвійович Фінк (1910 — 1988)

Найбільш простим методом декодування згорткового коду, який вперше запропонував Уозенкрофт, а більш ефективну, модифіковану версію цього алгоритму, трішки пізніше розробив Фано, є алгоритм послідовного декодування [33]. Розглянемо цей алгоритм на прикладі кодеру з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$, структурна схема якого наведена на рис. 3.61. Як і у прикладі, який розглядався у підрозділі 3.8.5, будемо вважати, що на вхід подана послідовність **m** = 1 1 0 1 1 та кодер формує вихідну послідовність **U** = 1101010001. Проаналізуємо випадок, коли прийнята кодова послідовність **Z** є спотвореним кодовим словом. Декодер згорткового коду, який працює за принципом послідовного декодування, має копію кодового дерева, яке еквівалентне зображеному на рис. 3.65.

Узагальнена блок-схема описаного алгоритму послідовного декодування згорткового коду наведена на рис. 3.72.

В цілому алгоритм послідовного декодування є аналогічним способу розв'язування задачі декодування, розглянутому у прикладі 3.60. Декодер починає аналіз дерева з моменту часу t_1 та генерує обидві можливі шляхи, які виходять з першого вузла. Якщо n перших символів, які прийняті, співпадають з будь-яким з обраних шляхів, декодер слідує безпосередньо за цим шляхом. Якщо узгодження між вхідним сигналом та поточним станом декодеру не існує, він слідує за найбільш імовірним шляхом, для якого кодова відстань між очікуваною та прийнятою кодовою комбінацією є мінімальною.

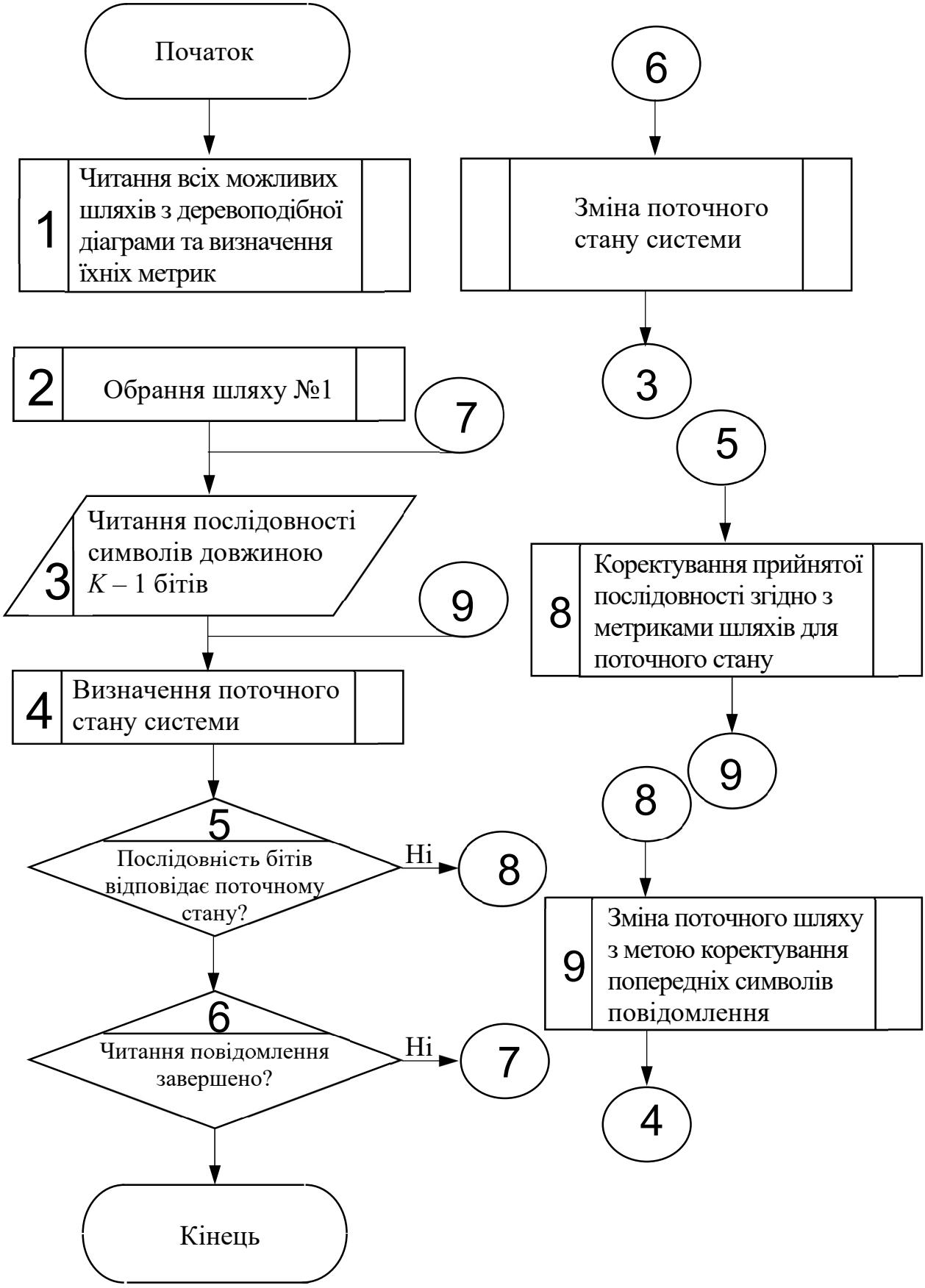


Рис. 3.72 Узагальнена блок-схема алгоритму послідовного декодування згорткового коду

Якщо дві гілки дерева є рівноможливими, декодувальний пристрій робить довільний вибір однієї з них. Подальший аналіз прийнятої послідовності за структурою кодового дерева повинен, в решті решт, показати, який вибір є правильним, а який хибним. На кожному рівні дерева декодер аналізує нові гілки та порівнює їх із попередніми прийнятими кодовими комбінаціями. Пошук за цим методом продовжується до тих пір, доки все дерево не буде пройдено за найбільш імовірним шляхом, при цьому підраховується загальна кількість співпадань. Якщо кількість неспівпадань перевищує наперед задане число, декодер вважає, що він знайшов неправильний шлях, відкидає його та починає пошук з самого початку. Також декодер має пам'ять списку шляхів, які відкидаються, що дозволяє уникнути можливості їх повторення під час наступного проходження дерева. Тобто, послідовне декодування згорткового коду ґрунтується на комбінаторному перебиранні всіх можливих шляхів дерева без використання будь-яких алгоритмів оптимального пошуку [33]. Головні теоретичні відомості з основ комбінаторного аналізу були розглянуті у першому томі другої частини посібника [48].

Припустимо, що четвертий та сьомий символ прийнятої кодової послідовності **U** є помилковими, тобто, прийнята спотворена послідовність **Z** = 1100011001.

Для розв'язування цієї задачі необхідно проаналізувати структуру дерева, наведену на рис. 3.65, результат такого аналізу показаний на рис. 3.73. Алгоритм цього аналізу ґрунтується на загальній теорії деревоподібних структур, яка є одним із важливих розділів дискретної математики та розглядалася у першому томі другої частини посібника [48].

Спочатку необхідно задати критерій аналізу, за яким будемо визначати вагу гілок дерева. Цей критерій пов'язаний із критичною величиною кількості неспівпадань між прийнятою та можливою кодовою комбінацією, за умови досягнення якої здійснюється повернення назад за структурою дерева та пошук можливого іншого шляху. Припустимо, що цій умові відповідає значення $j = 3$, де j – максимальна можлива кількість неспівпадань. Відповідні значення j для всіх гілок шляху, який аналізується, вказані на рис. 3.73. Розглянемо цей аналіз структури дерева більш досконало [33].

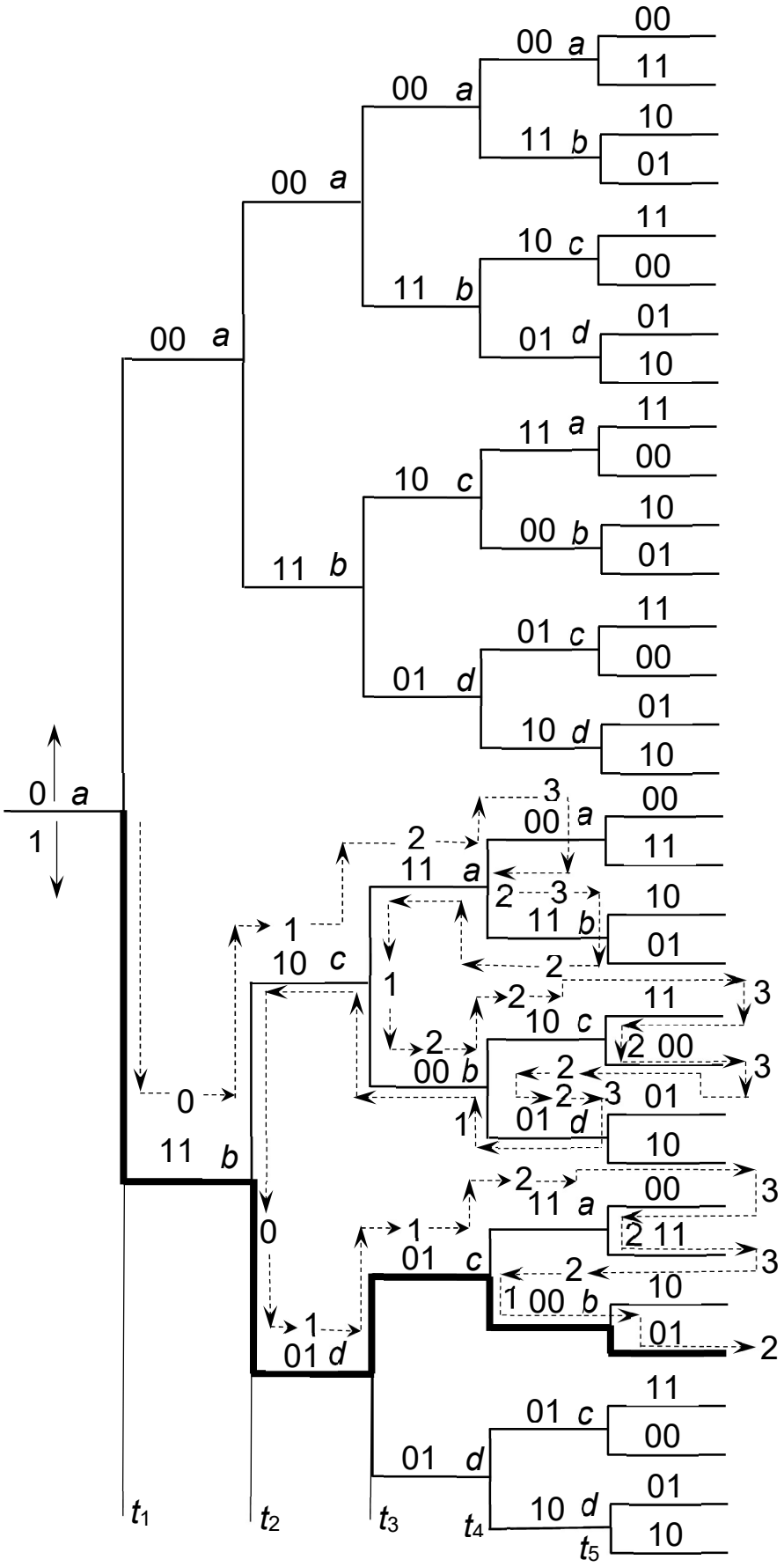


Рис. 3.73 Пояснення алгоритму послідовного декодування згорткового коду на основі аналізу деревоподібної діаграми

1. У момент часу t_1 декодер приймає послідовність символів 11 та порівнює їх із кодовими словами для двох гілок, які виходять з першого вузла дерева.

2. На цьому першому етапі одна з гілок, яка спрямована вниз, цілком відповідає прийнятій кодовій комбінації, тому можна вважати цю гілку правильною та продовжувати аналіз прийнятої кодової комбінації далі.

3. У момент часу t_2 декодер приймає послідовність символів 00 та порівнює її з можливими кодовими словами для цього вузла дерева, а це кодові слова 10 та 01.

4. Обидва шляхи у даному випадку не співпадають з прийнятою кодовою комбінацією, тобто виявлена помилка декодування. Інша проблема полягає у тому, що для обох можливих шляхів кодова відстань між прийнятою комбінацією та кодовими словами складає 1, тобто критерій вибору правильного шляху за мінімальною різницею між прийнятою та можливою кодовою комбінацією також не спрацьовує. За такої умови декодер обирає один з можливих шляхів, якому відповідає кодово слово 01. Крім цього, вмикається лічильник неспівпадань, вміст якого набуває значення 1.

5. У момент часу t_3 декодер приймає послідовність символів 01 та порівнює її з можливими кодовими словами для поточного вузла дерева, а це кодові слова 11 та 00.

6. Тут також, як і в пункті алгоритму 4, жодне з кодових слів не відповідає вхідній кодовій комбінації, а кодова відстань між прийнятою комбінацією та обома можливими варіантами складає 1. Тому на цьому етапі декодер знову здійснює довільний вибір гілки дерева, якій відповідає кодове слово 11, а вміст лічильника неспівпадань набуває значення 2.

7. У момент часу t_4 декодер приймає послідовність символів 10 та порівнює її з можливими кодовими словами для поточного вузла дерева, а це кодові слова 00 та 11.

8. Тут також, як і в пунктах алгоритму 4 та 6, знову жодне з кодових слів не відповідає вхідній кодовій комбінації, а кодова відстань між прийнятою комбінацією та обома можливими варіантами складає 1. Тому на цьому етапі декодер, як і раніше, здійснює довільний вибір гілки дерева, якій відповідає кодове слово 00, а вміст лічильника неспівпадань набуває значення 3.

9. Оскільки лічильник кількості неспівпадань прийняв критичне значення 3, необхідно повернутися назад за структурою дерева та шукати інші можливі шляхи.

10. Альтернативний шлях на четвертому рівні відповідає кодовому слову 11, а прийнята кодова послідовність становить 10. Тому, прийнявши цей шлях, декодер відразу збільшує вміст лічильника неспівпадань на 1.

11. Оскільки вміст лічильника 3 тепер знову є критичною величиною, проаналізований альтернативний шлях також вважається хибним та відхиляється.

12. Оскільки на рівні t_4 всі шляхи проаналізовані та жодний з них не є правильним, декодер переходить на рівень t_3, а вміст лічильника неспівпадань зменшується та набирає значення 1.

13. У вузлі t_3 декодер обирає невикористаний шлях, якому відповідає кодове слово 00. Знову має місце неспівпадання, тому лічильник неспівпадань приймає значення 2.

14. У вузлі t_3 декодер обирає гілку дерева, якій відповідає кодове слово 00, оскільки це значення співпадає з прийнятим. За такої умови лічильник неспівпадань зберігає своє значення 2, а декодер переходить до аналізу стану системи у вузлі t_5.

15. У вузлі t_5 гілкам дерева відповідають кодові слова 00 та 11, а прийнята кодова комбінація 01. За такої умови жодна з гілок дерева не має переваг, тому декодер навмання обирає гілку, якій відповідає кодове слово 11.

16. Лічильник неспівпадань приймає значення 3. За такої умови декодер

повертається назад до вузла t_5 та обирає іншу гілку дерева. Тоді значення лічильника неспівпадань зменшується на 1, тобто, кількість неспівпадань становить 2.

17. У вузлі t_5 необраною залишилась лише одна гілка, якій відповідає кодове слово 00. Знову має місце неспівпадання та значення лічильника неспівпадань тепер становить 3.

18. Оскільки кількість неспівпадань у вузлі t_5 за обома гілками має критичне значення, декодер залишає цей шлях. За такої умови значення лічильника неспівпадань зменшується на 1 та становить 2, а декодер шукає альтернативний шлях з вузла t_4 до вузла t_5.

19. Оскільки декодер здійснив повернення назад до вузла t_4, значення лічильника неспівпадань зменшується на 1 та становить 1.

20. У вузлах t_4 та t_3 всі шляхи вже перевірені та помічені декодером як хибні, тому декодер робить ще два одне повернення назад до вузла t_2. За такої умови лічильник неспівпадань набуває значення 0.

21. У вузлі t_2 декодер обирає альтернативну гілку дерева, якій відповідає кодове слово 01. Знову має місце неспівпадання із прийнятою кодовою комбінацією 00. Лічильник кількості неспівпадань за такої умови приймає значення 1.

22. Декодер переходить до вузла t_3. Прийната у цей момент кодова комбінація становить 01, і вона співпадає із кодовим словом однієї з гілок дерева. За такої умови лічильник кількості неспівпадань зберігає своє значення 1, а декодер переходить до вузла t_4.

23. У вузлі t_4 кодове дерево має дві гілки, яким відповідають кодові слова 11 та 00, а прийнята кодова комбінація становить 10. Оскільки жодна з гілок дерева не є пріоритетною, спочатку декодер навмання обирає гілку із вагою 11. За такої умови лічильник кількості неспівпадань набирає значення 2, а декодер переходить до вузла t_5.

24. У вузлі t_5 кодове дерево має дві гілки, яким відповідають кодові слова 11 та 00, а прийнята кодова комбінація становить 01. Для обох з цих гілок співпадання із прийнятою кодовою комбінацією 01 не існує, тому лічильник неспівпадань набирає значення 3. За такої умови декодер повертається до вузла t_4, а значення лічильника кількості неспівпадань зменшується до 1.

25. У вузлі t_4 декодер обирає альтернативну гілку із вагою 00. Тоді лічильник кількості неспівпадань набирає значення 2, а декодер переходить до вузла t_5.

26. У вузлі t_5 однієї з гілок кодового дерева відповідає кодове слово 01, яке співпадає зі значенням прийнятої кодової на цей момент комбінації.

27. Оскільки декодер дійшов до кінця дерева, процес декодування вважається закінченим. В результаті декодування отримана кодова комбінація 1101010011, яку можна прочитати за гілками дерева, через які пройшов сформований шлях. Кількість помилок, згідно із лічильником кількості неспівпадань, складає 2.Тобто, згідно із поставленим завданням та внесеними у код помилками, можна вважати, що воно розв’язане правильно, оскільки початкова, безпомилкова кодова комбінація поновлена та виявлена кількість помилок у коді.

Розглянутий алгоритм послідовного декодування згорткового коду, узагальнена блок-схема якого наведена на рис. 3.72, також був реалізований у прикладі 3.60.

3.8.6.2 Декодування із зворотним зв’язком

У декодері згорткового коду із зворотним зв’язком реалізована схема прийняття рішень, яка базується на аналізі метрик шляху попередніх гілок кодового дерева. Принцип роботи декодера із зворотним зв’язком є досить простим, головна проблема полягає у складності реалізації такого декодера [33].

Для формування рішення про правильну кодову комбінацію у декодері

із зворотним зв’язком використовують інформаційні біти в розряді j, виходячі з метрик шляхів, отриманих із розрядів $j, j + 1, \ldots, j + m$, де m – наперед задане додатне ціле число. Важливим парметом декодера із зворотним зв’язком є також довжина упередження L (англійський термін – look-ahead lenth), який визначається через значення m як

$$L = m + 1. \tag{3.308}$$

З фізичної точки зору довжина упередження являє собою кількість прийнятих кодових символів, яка виражається через відповідну кількість вхідних бітів, задіяних для декодування інформаційного біту.

Сутність декодування із зворотним зв’язком полягає у тому, що рішення про значення інформаційного біту, 0 або 1, приймається залежно від того, на якому шляху у вікні упередження (англійський термін – look-ahead window) відстань Хеммінга є мінімальною. Вікно упередження являє собою послідовність прийнятих кодових комбінацій від такту t_j до t_{j+m}. Тобто, кожного разу обирається такий шлях, для якого кодова відстань між прийнятою та передбаченою комбінаціями є мінімальною.

Розглянемо приклад роботи декодеру згорткового коду із зворотним зв’язком на основі кодового дерева для згорткового коду з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$. Структурна схема такого кодеру наведена на рис. 3.61, а деревоподібна діаграма декодера – рис. 3.65.

Будемо вважати, що $L = 3$. Починаючи з першої гілки, декодер із зворотним зв’язком обчислює $2^L = 8$ сукупних метрик шляхів за Хеммінгом та вирішує, яким є прийнятий біт, наступним чином. Якщо шлях з мінімальною відстанню за Хеммінгом розташований на верхній гілці дерева, тоді прийнятий біт дорівнює 0, а у противному випадку – 1.

Наприклад, будемо вважати, що прийнята кодова послідовність становить $\mathbf{Z} = 1101010001$. Розглянемо 8 шляхів від моменту часу t_1 до моменту часу t_3. Частина дерева, яка відповідає цьому проміжку часу, на

рис. 3.74 позначена літерою *A*. Згідно з алгоритмом декодування із зворотним зв'язком необхідно обчислити 8 метрик, 4 з яких відповідають верхній, а 4 – нижній частині дерева. Відповідні розрахунки дають результати, які наведені у таблиці 3.28.

Таблиця 3.28 – Метрики шляхів для частини дерева, яка відповідає блоків *A* та *B*

Метрики гілок для шляхів верхньої та нижньої частини дерева					Сумарне значення	Блок
Метрики верхньої частини дерева	4	2	0	6	6	*A*
Метрики нижньої частини дерева	0	0	0	0	0	
Метрики верхньої частини дерева	2	2	4	4	12	*B*
Метрики нижньої частини дерева	0	0	0	0	0	

Зрозуміло, що для блоку *A* меншій метриці відповідає нижня частина дерева, відповідно до цього першим бітом послідовності, яка декодуєтсья, є 1. На наступному кроці нижня частина дерева розшпряється на 1 крок, а вузол t_1 відкидається. Таким чином, розглядаються вузли від t_2 до t_4. Отримавши, таким чином, частину структури дерева, яку необхідно розглянути на другій ітерації, виділяємо блок *B*, відповідна операція також відображена на рис. 3.74. Обчислимо метрики цього блоку для прийнятої кодової комбінації, відповідні значення для блоку *B* також наведені у таблиці 3.28. Далі декодер із зворотним зв'язком рухається за описаним ітераційним шляхом та обирає в кожному вузлі верню або нижню гілку дерева. Відповідні операції процесу декодування показані на рис. 3.74.

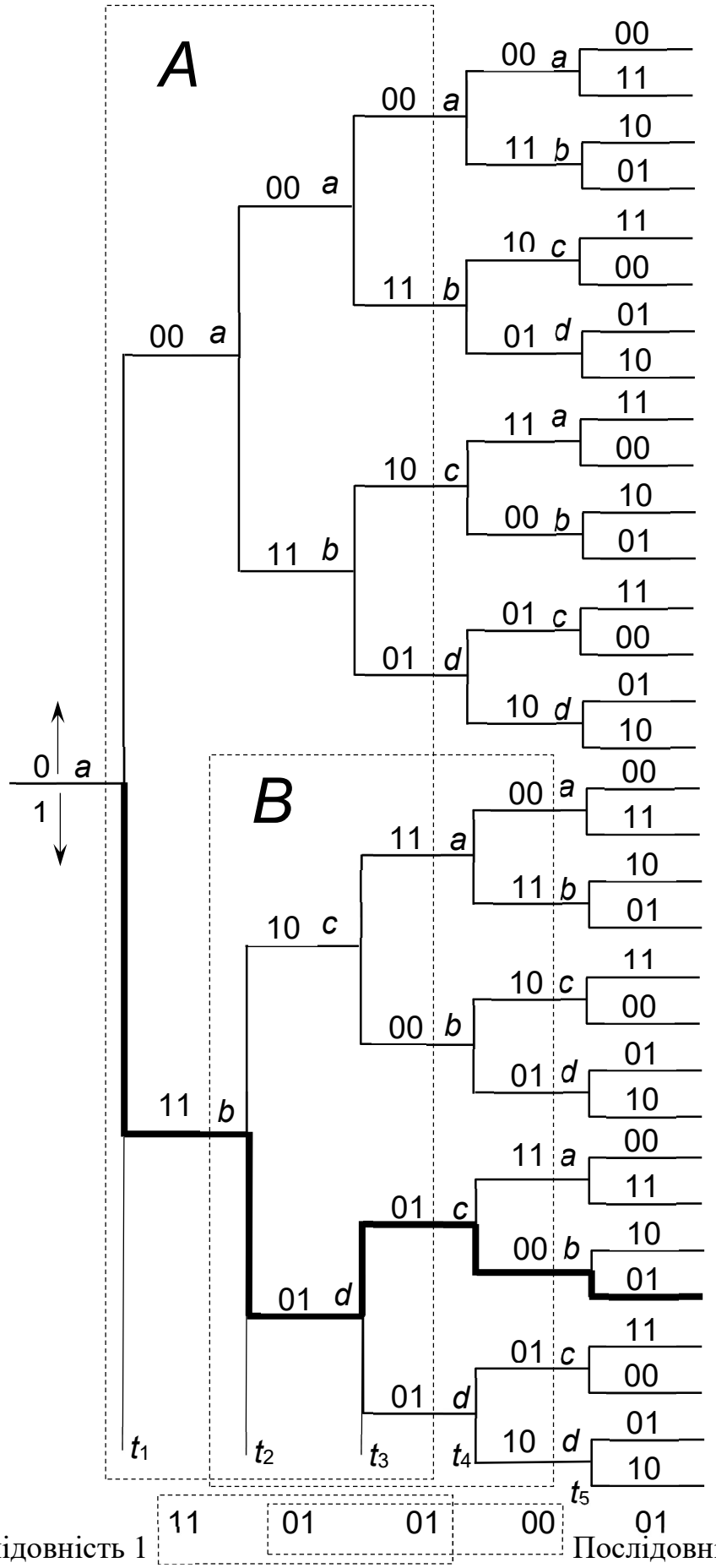


Рис. 3.74 Ілюстрація принципу роботи декодеру зі зворотним зв’язком

3.8.6.3 Декодування за принципом максимальної правдоподібності та алгоритм Вітербі

Декодування за принципом максимальної правдоподібності є найбільш ефективним способом аналізу послідовностей згорткового коду та пошуку спотворених символів у ньому. Цей спосіб декодування розглядається для симетричного двійкового каналу, відповідні теоретичні відомості та структурна схема цифрової системи зв'язку з симетричним каналом були наведені у першій частині цього навчального посібника [1].

Відомо, що симетричний двійкових канал можна описати за допомогою таких умовних імовірностей [33]:

$$P(0|1) = P(1|0) = p; \qquad P(1|1) = P(0|0) = 1 - p. \tag{3.309}$$

З фізичної точки зору співвідношення (3.309) можна пояснити наступним чином. У симетричному двійковому каналі зв'язку імовірність того, що вихідний сигнал буде відрізнятися від вхідного дорівнює p, а, відповідно, імовірність протилежної події, яка полягає в тому, що вхідний та вихідний сигнали будуть співпадати, складає $1 - p$.

У теорії згорткових кодів розрізняють системи з м'якою та жорсткою системами прийняття рішень. Ці системи відрізняються тим, що жорстка система відповідає двійковим сигналам та має на виході лише два можливих рівні, у той час як м'яка система є, хоча й цифровою, але багаторівневою [33].

Зрозуміло, що система декодування для симетричного двійкового каналу може бути лише жорсткою.

Припустимо, що, як і в попередніх прикладах, переданою була кодова послідовність $\mathbf{U}$, а замість неї прийнята послідовність $\mathbf{Z}$. Через d_m позначмо кодову відстань між прийнятою послідовністю $\mathbf{Z}$ та послідовність $\mathbf{U}^{(m)}$, яку кодер обирає замість спотвореної послідовності $\mathbf{Z}$. За принципом максимуму правдоподібності декодер обирає таку послідовність $\mathbf{U}^{(m)}$, для якої значення d_m є мінімальним.

Загальні математичні основи теорії правдоподібності, а також можливості

практичного використання цієї теорії для статистичної оцінки параметрів цифрових та аналогових сигналів, були описані у навчальному посібнику [49]. Тепер використаємо ці теоретичні оцінки та практичні рекомендації для імовірнісного аналізу параметрів згорткового коду.

Згідно з записаними співвідношеннями (3.309) імовірність того, що переданою послідовністю була саме послідовність $\mathbf{U}^{(\mathbf{m})}$, становить [33]:

$$p\left(\boldsymbol{Z}|\mathbf{U}^{(\boldsymbol{m})}\right) = p^{d_m}(1-p)^{L-d_m}. \tag{3.310}$$

де L – довжина кодової комбінації.

Більш простою формою запису співвідношення (3.310) є логарифмічна форма, яка дозволяє перейти від нелінійної степеневої залежності до відповідної лінійної. Для цього випадку співвідношення (3.310) можна переписати наступним чином:

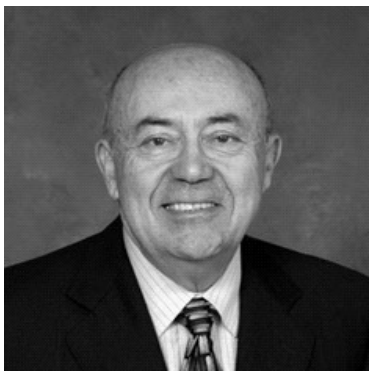
$$\lg\left(p\left(\boldsymbol{Z}|\mathbf{U}^{(\boldsymbol{m})}\right)\right) = -d_m \lg\left(\frac{1-p}{p}\right) + L\lg(1-p). \tag{3.311}$$

Зрозуміло, що в отриманому співвідношенні (3.311) значення $\lg\left(\frac{1-p}{p}\right)$ та $L\lg(1-p)$ є постійними величинами, тобто, його можна переписати у спрощеному вигляді:

$$\lg\left(p\left(\boldsymbol{Z}|\mathbf{U}^{(\boldsymbol{m})}\right)\right) = -Ad_m + B. \tag{3.312}$$

Отриманий результат та співвідношення (3.312) з практичної точки зору означають, що максимально правдоподібною є така кодова комбінація, для якої кодова відстань між комбінаціями $\mathbf{U}^{(\boldsymbol{m})}$ та $\mathbf{Z}$ є мінімальною.

Перший ефективний алгоритм декодування згорткового коду, оснований на принципу правдоподібності, розробив в 1967 р. американський математик французького походження Ендрю Вітербі [33]. Сутність алгоритму Вітербі полягає у тому, що в основу його проложений принцип максимальної правдоподібності та наведене співвідношення (3.312). Але суттєва відмінність алгоритму Вітербі від простого комбінаторного перебирання всіх шляхів дерева полягає у тому, що кількість шляхів, які аналізуються, може бути суттєво знижена через аналіз ґратки коду із заданими параметрами.



Ендрю Вітербі
Нар. 1935

Джиммі К. Омура
Нар. 1940

В основу алгоритму Вітербі покладено обчислення міри подібності між сигналом, який прийнятий в момент часу t_i, та усіма шляхами гратки, які входять до кожного стану у цей момент часу. Тобто, ті шляхи гратки, які достовірно не можуть бути оптимальними за принципом максимальної правдоподібності та формулою (3.312), взагалі не розглядаються. Якщо до одного стану входять два шляхи, тоді обертається той з них, який має кращу метрику, і такий шлях називається, шляхом, що виживає [33]. Оскільки шлях, що виживає, обирається для кожного стану системи, декодер, через заглиблення до структури гратки, обирає такий шлях, який є найбільш імовірним для прийнятої кодової комбінації. Тобто, попереднє відхилення малоймовірних шляхів за алгоритмом Вітербі суттєво спрощує аналіз структури дерева коду.

Е. Вітербі, запропонувавши свій алгоритм, аналізував міру подібності між прийнятим сигналом та структурою дерева, але у своїх міркуваннях він не використовував принцип максимуму правдоподібності. В 1969 р. американський математик японського походження Д.К. Омура вперше показав, що теоретичним підґрунтям алгоритму декодування Вітербі дійсно є принцип максимуму правдоподібності. Тому в літературі задачу відбору оптимального шляху розглядають по різному, і як обрання шляху з максимальною метрикою правдоподібності, і як обрання кодового слова з мінімальною метрикою відстані [33]. Перший підхід відповідає теоретичним міркуванням Д.К. Омури, а другий – загальній концепції Е. Вітербі. Зв’язок

між цими двома підходами дають співвідношення (3.309 – 3.312).

Розглянемо алгоритм побудови згорткового коду, який базується на застосуванні ґраткових діаграм, для чого знову звернемося до рис. 3.66, де зображена ґраткова діаграма кодера з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$ [33]. На рис. 3.75 на основі цієї ґраткової діаграми показана послідовність символів вхідного та виходного сигналів у різні моменти часу.

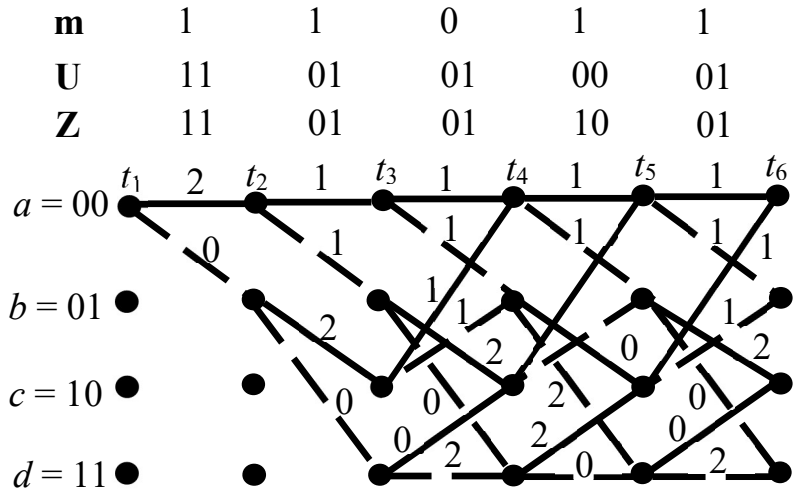


Рис. 3.75 Ґраткова діаграма для формування згорткового коду, наведена на рис. 3.66, з позначеними кодовими відстонями між прийнятою та очікуваною комбінаціями

Проаналізуємо ґраткову діаграму, яка наведена на рис. 3.75. Будемо вважати, що відстані між кодовами комбінаціями відповідають структурі згорткового кода з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$. Відповідно до алгоритму Вітербі першим кодовим словом буде 00. Після надходження нового символу кожна гілка кодової ґратки помічається позначкою подібності (англійський термін – similarity mark) [33]. Кодова відстань між послідовністю символів 00, яка прогнозується, і послідовністю 11, що надійшла, складає 2. У зв'язку з цим відрізок верхньої гілки ґраткової діаграми між точками t_1 та t_2 помітимо цифрою 2. Розглянемо тепер нижнюю гілку. Тут стан 01 точно відповідає прийнятому кодовому слову 11, відповідно, кодова відстань для цієї гілки дорівнює 0. Аналогічно визначаються кодові відстані для всіх інших гілок ґраткової діаграми, відповідні цифри проставлені на рис. 3.75.

Тепер необхідно визначити сумарну метріку шляху за Хеммінгом, що можна легко зробити через сумування відповідних метрік гілок ґраткової діаграми, через які прокладено відповідний шлях. Тобто, насамперед слід розглянути всі можливі шляхи між точками t_1 та t_5. Таких шляхів можна знайти 4, і всі вони показані на рис. 3.76. Там же вказані відповідні метріки гілок, які вже були позначені раніше на рис. 3.75.

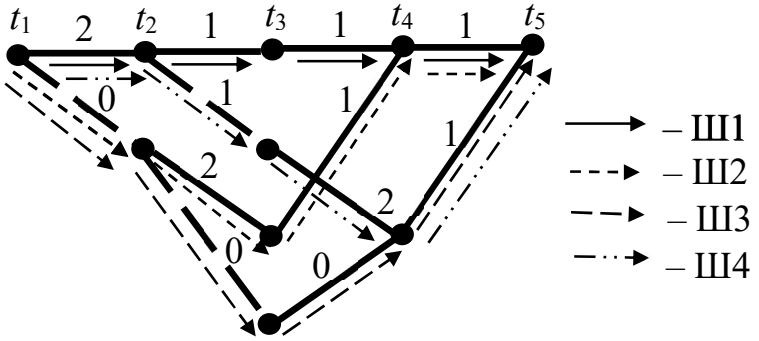


Рис. 3.76. Ілюстрація принципу вибору шляху з найменшою метрикою за Хеммінгом на ґратковій діаграмі за алгоритмом Вітербі

Метрики кожного з можливих чотирьох шляхів становлять.

Ш1: 2 + 1 + 1 + 1 = 5; Ш2: 0 + 2 + 1 + 1 = 4;

Ш3: 0 + 0 + 0 + 1 = 1; Ш4: 2 + 1 + 2 + 1 = 6.

Сутність побудови згорткового коду Вітербі полягає у тому, що під час оптимізації ґраткової діаграми шляхи, які мають більші метрики, послідовно видаляються. Тобто, передбачається, що між двома станами кодера можливий лише один шлях, який має найменшу метріку. Наприклад, для приклада, який розглядається, будемо вважати, що стан $a = 00$ у момент часу t_5 може бути досягнутим тільки через шлях Ш3. Тому саме цей шлях під час оптимізації ґраткових діаграм називають шляхом виживання (англійський термін – way of survival). Зрозуміло, що такий підхід до оптимізації діаграм відповідає теорії процесів Маркова, описаних у другому томі другої частини цього посібника [49]. Теоретичним підґрунтям для цього піддходу є також принцип максимума правдоподібності та на теорія систем штучного інтелекту, які були описані у другому [49] та третьому [50] томах другої частини цього посібника. Зрозуміло, що принцип максимуму правдоподібності відповідає наведеним

вище співвідношенням (3.309 – 3.312). Після вибору оптимального шляху всі інші гілки ґраткової діаграми видаляються, і така ґраткова діаграма називається оптимізованою за Вітербі. Оптимізована за Вітербі діаграма, еквівалентна діаграмі, наведеній на рис. 3.76, зображена на рис. 3.77. Тут через Г позначені метріки станів, які цілком відповідають метрикам шляхів, які обрані.

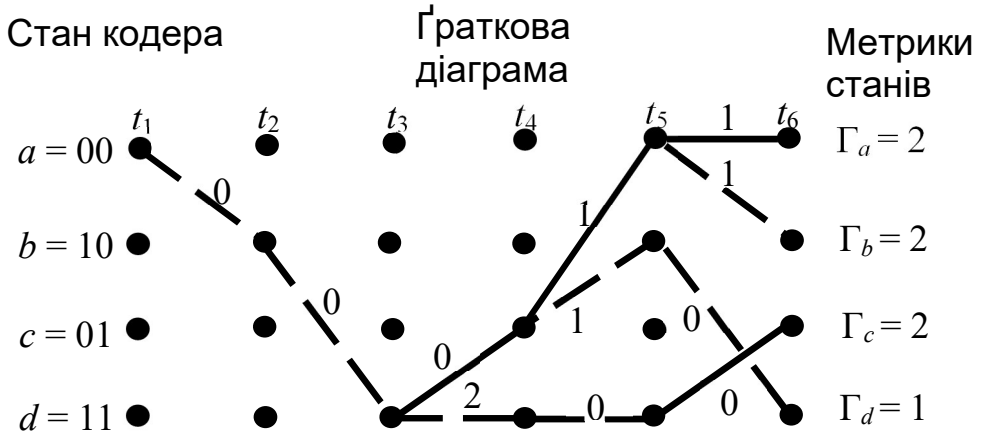


Рис. 3.77. Ґраткова діаграма, оптимізована за Вітербі

Розглянемо логіку побудови алгоритму декодування Вітербі більш досконало. У кожний момент часу t_i у ґратці існує 2^{K-1} станів, де K – парааметр коового обмеження. Тобто, до кожного стану може входити два шляхи [33]. Завдання декодування за Вітербі полягає в тому, що необходно обчислити метрики цих шляхів за Хеммінгом та виключити із розгляду ті з них, для яких метрика є більшою. Такі обчислення проводяться для кожного з 2^{K-1} вузлів, у заданий момент часу t_i. Після цього декодер переходить до наступного моменту часу t_{i+1} та обчислювальний процес продовжується. На даний момент часу метрика шляху, який виживає для кожного стану, позначається як метрика цього стану для поточного часу. В алгоритмі Вітербі, як і в алгоритмі послідовного декодування, якщо на даній часовій ітерації дві гілки шляху, одна з яких є правильною, а інша – хибноб, мають однакову вагу, одно з цих гілок обирається виапдковим чином. Якщо ця гілка обрана помилково, корекція вибору здійснюється на анступних ітераціях [33].

Загалом алгоритм Вітербі у значній мірі є схожим на алгоритм

послідовного декодування, описаний у підрозділі 3.8.6.1. Суттєва відмінність алгоритму Вітербі полягає в тому, що аналіз станів декодера у попередні моменти часу в даному разі не здійснюється. Але і в цьому випадку такий аналіз є потрібним, проте він реалізований в непрямій формі через обчислення метрик шляхів до поточного стану. Річ у тому, що шляхи із найменшою метрикою для станів у моменти часу t_i та t_{i+j} можуть суттєво відрізнятися, і саме за рахунок цього в алгоритмі Вітербі реалізований аналіз найбільш правдоподібних станів декодувальної системи в попередні моменти часу. Якщо під час передавання інформації виникла помилка та через цю помилку в момент часу t_i обраний неправильний шлях, у деякий наступний момент часу t_{i+j} буде обраний інший шлях, якому тепер відповідає найменша метрика, і у такий спосіб помилка передавання інформації буде виправлена.

Проаналізуємо більш досконало всі кроки алгоритму декодування Вітербі для конкретного прикладу, який розглядається. Нехай кодові послідовності **m**, **U** та **Z** аналогічні наведеним на рис. 3.75. Для початку припустимо, що декодер знає правильний початковий стан ґратки. Насправді, таке припущення не є необхідним, але у разі його прийняття подальший аналіз станів ґратки суттєво спрощується. Згідно з заданою кодовою послідовністю **Z**, в момент часу отримані t_1 кодові символи 11, які відповідають вхідному біту 1. Із аналізу структури ґратки, наведеної на рис 3.75, видно, що з початкового стану системи 00 можна перейти або до стану a = 00, або до стану b = 01, відповідна структура цієї частини дерева відображена на рис. 3.78, а. Обчислимо метрики за Хеммінгом для цих двох гілок дерева ґратки у цей моменту часу за умови відомого сигналу послідовності **Z** 11. Для гілки, якій відповідає перехід 00 → 00, метрика за Хеммінгом дорівнює 2, а для гілки, якій відповідає перехід 00 → 00, ця метрика становить 0. На початку аналізу структури дерева обидві гілки зі стану a = 00 вважаються можливими та не відкидаються [33], що також наочно видно з рис. 3.78, а. Далі ґраткова діаграма розгалужується, тому в момент часу t_2 з кожної вершини дерева виходять ще по дві гілки. Відповідна структура дерева показана на рис 3.78, б.

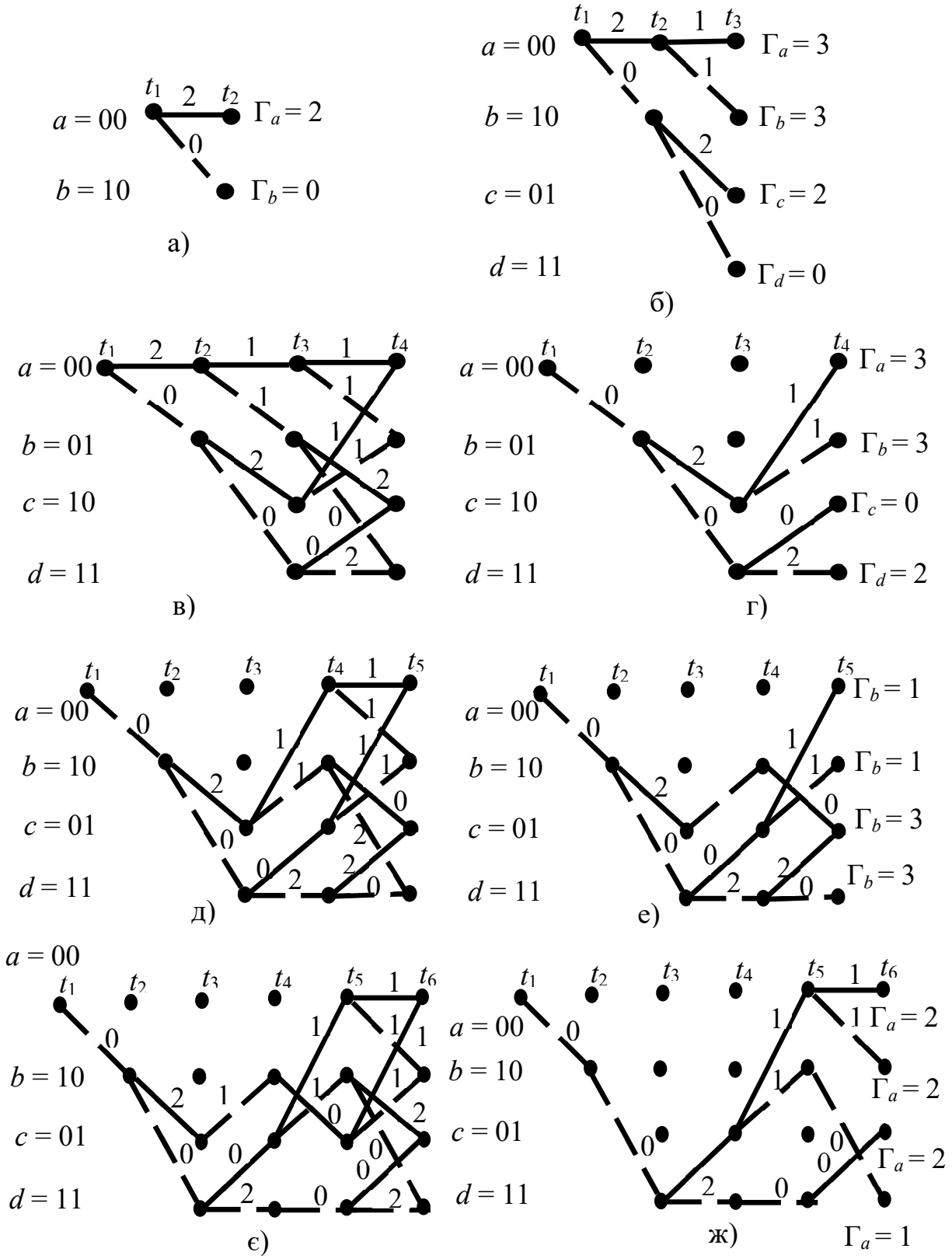


Рис. 3.78 Обрання шляхів, які виживають, для ґраткової діаграми згорткового коду, показаної на рис. 3.75. а) – виживання на момент часу t_2; б) – виживання на момент часу t_3; в) – порівняння метрик на момент часу t_4; г) – виживання на момент часу t_4; д) – порівняння метрик на момент часу t_5; е) – виживання на момент часу t_5; є) – порівняння метрик на момент часу t_6; ж) – виживання на момент часу t_6

На рис. 3.78, б, також позначені метрики шляхів, які ведуть до станів *a*, *b*, *c* та *d* у момент часу t_3 та позначені Γ_a, Γ_b, Γ_c, та Γ_d відповідно. Оскільки помилки передавання інформації на цьому етапі не визначено, існує шлях до стану *d*, метрика якого $\Gamma_d = 0$. Саме цей шлях на момент часу t_3 і вважається найбільш правдоподібним. Надалі, на рис. 3.28 також позначені метрики шляхів після аналізу ґратки у відповідні моменти часу та визначення шляхів, які необхідно видалити на даній ітерації алгоритму Вітербі. Видно, що ілюстрації цієї операції відповідають рис. 3.78 г, е та ж.

На рис.3.78, в, показана структура ґратки до моменту часу t_4. Тут також до станів *a*, *b*, *c* та *d* ведуть чотири шляхи, їх аналіз та обчислення метрик за Хеммінгом проведено на рис. рис.3.78, г. Інші можливи шляхи, які ведуть до станів *a*, *b*, *c* та *d*, у момент часу t_4, видалені. Як видно з рис. 3.78, г, у момент часу t_4 найменші метрики шляхів до чотирьох станів системи складають $\Gamma_a = 3$, $\Gamma_b = 3$, $\Gamma_c = 0$ та $\Gamma_d = 2$. Цілком аналогічно декодером проводиться аналіз можливих станів системи у моменти часу t_5 та t_6, відповідна структура ґратки та метрикі шляхів, які виживають, показані на рис. 3.78, д – ж. Зрозуміло, що на кожній часовій ітерації до кожного із станів системи існують два шляхи, і той шлях, який має більшу метрику за Хеммінгом, згідно з алгоритмом декодування Вітербі видаляється.

Незважаючи те, що алгоритм декодування Вітербі є досить простим та досконалим, він має і один суттєвий недолік. Під час декодування реальних згорткових кодів іноді виникають певні труднощі у випадку видалення з ґраткової діаграми шляху із нульовою метрикою, який характеризує безпомилкове передавання інформації. Більш досконало головні принципи формування кодів Вітербі описані у підручнику [33]. З іншого боку, видалення у ґратці шляхів із високою метрикою завжди гарантує, що в оптимізованій ґратці у будь-який момент часу шляхів не буде більше, ніж станів системи. На рис. 3.78 б, г, е та ж чітко видно, що, незалежно від часу, до кожного зі станів, *a*, *b*, *c* та *d*, існує лише один шлях, оскільки зайві шляхи видаляються під час оптимізації ґратки.

3.8.6.4 Порівняльний аналіз алгоритмів Вітербі та послідовного декодування

Проведемо порівняльний аналіз алгоритмів послідовного декодування та Вітербі, описаних у підрозділах 3.8.6.1 та 3.8.6.3. Головний недолік алгоритму декодування Вітербі полягає в тому, що якщо імовірність появи помилки експоненціально спадає зі збільшенням параметру кодового обмеження K, проте кількість станів декодера, та, відповідно, його складність, експоненціально зростає із збільшенням цієї величини [33]. Проте алгоритм декодування Вітербі має іншу важливу перевагу. Вона полягає у тому, що обчислювальна складність цього алгоритму не залежить від характеристик каналу зв'язку та співвідношення потужності сигналу до потужності шуму [1]. На відміну від цього, у разі використання послідовного декодування асимптотично досягається та сама імовірність появи помилки, що й для принципу максимальної правдоподібності, але без необхідності перебирання всіх можливих станів декодера. Фактично, за умови використання алгоритму послідовного декодування, кількість станів системи, які перебираються, суттєво не залежить від довжини кодового обмеження K, що дає можливість реалізовувати ефективні системи кодування з великим значенням цього параметру, наприклад, $K = 41$ [33]. А, як було відмічено вище, збільшення значення параметру кодового обмеження дозволяє значно зменшити імовірність виникнення під час передавання через канал зв'язку таких помилок кодових комбінацій, які неможливо виправити. Але алгоритм послідовного декодування має інший недолік. Він полягає в тому, що кількість метрик, які перебираються, є випадковою величиною. Тому, на відміну від алгоритму Вітербі, для послідовного декодування кількість хибних гіпотез та випадкових перебирань є функцією співвідношення потужності сигналу до потужності шуму η у каналі зв'язку. Цілком зрозуміло, що за умови високого значення η необхідно перебирати значно менше хибних гіпотез, ніж за умови низького значення цього параметру [33]. Такий недолік алгоритму послідовного декодування згорткового коду призводить до того, що обчислювальне навантаження декодеру та кількість станів, які необхідно зберігати в буфері пам'яті, суттєво залежить від режиму роботи каналу зв'язку та від співвідношення потужності сигналу до потужності шуму [1]. Зрозуміло,

що у разі збільшення потужності шуму у каналі зв'язку імовірність помилки збільшується та декодеру необхідно перебирати більшу кількість станів системи, і він буде робити це, доки не знайде найбільш імовірну гіпотезу. Найбільш складна ситуація в декодерах із послідовним декодуванням виникає за умови, коли середня швидкість передавання символів перевищує середню швидкість їх декодування. Такий випадок є стандартним граничним обмеженням стабільної роботи електронних систем, пов'язаним із фундаментальними законами теорії черг [49]. У разі виникнення такої ситуації буфер декодера, незалежно від його об'єму, в певний момент часу переповнюється, і тоді інформація, яка надійшла до нього, втрачається [33]. Тому сьогодні в декодерах згорткових кодів реалізують також інтелектуальні алгоритми послідовного декодування. Їх головна відмінність полягає в тому, що коли інформаційний потік, який надходить до буферу, перевищує кількість інформації, яка обробляється, система починає відкидати з буферу безпомилкові дані, а обчислювальний блок намагається шукати кодові комбінації з найменшими метриками [33]. Проте, як відмічалось у другому томі другої частини посібника, згідно з основним законом теорії черг електронні та комунікаційні системи, в яких вхідний інформаційний потік є перевантаженим відносно вихідного, завжди працюють нестабільно та таких ситуацій слід уникати [49]. Але для декодерів із послідовним декодуванням складність ситуації полягає в тому, що поріг перевантаження буфера суттєво залежить від параметра каналу η, тому важливим параметром для оцінки ефективності та стабільнотсі роботи таких декодувальних систем є обчислення імовірності переповнення буферу.

На рис. 3.79 показані типові криві, які відображують залежність P_B від значення співвідношення потужності сигналу до потужності шуму E_b / N_0 для різних способів формування згорткових кодів та декодування їхніх послідовностей [33]. З наведених графічних залежностей видно, що можливо досягти бітової помилки $P_B = 10^{-6}$ за умови E_b / N_0 = 8 дБ. Теоретична межа ефективного кодування за Шеноном складає E_b / N_0 = 11 дБ, тобто, ефективність сучасних електронних систем кодування наближається до теоретичної граничної величини. Способи теоретичних оцінок коректувальних параметрів згорткових кодів будуть розглянуті у підрозділі 3.8.13.

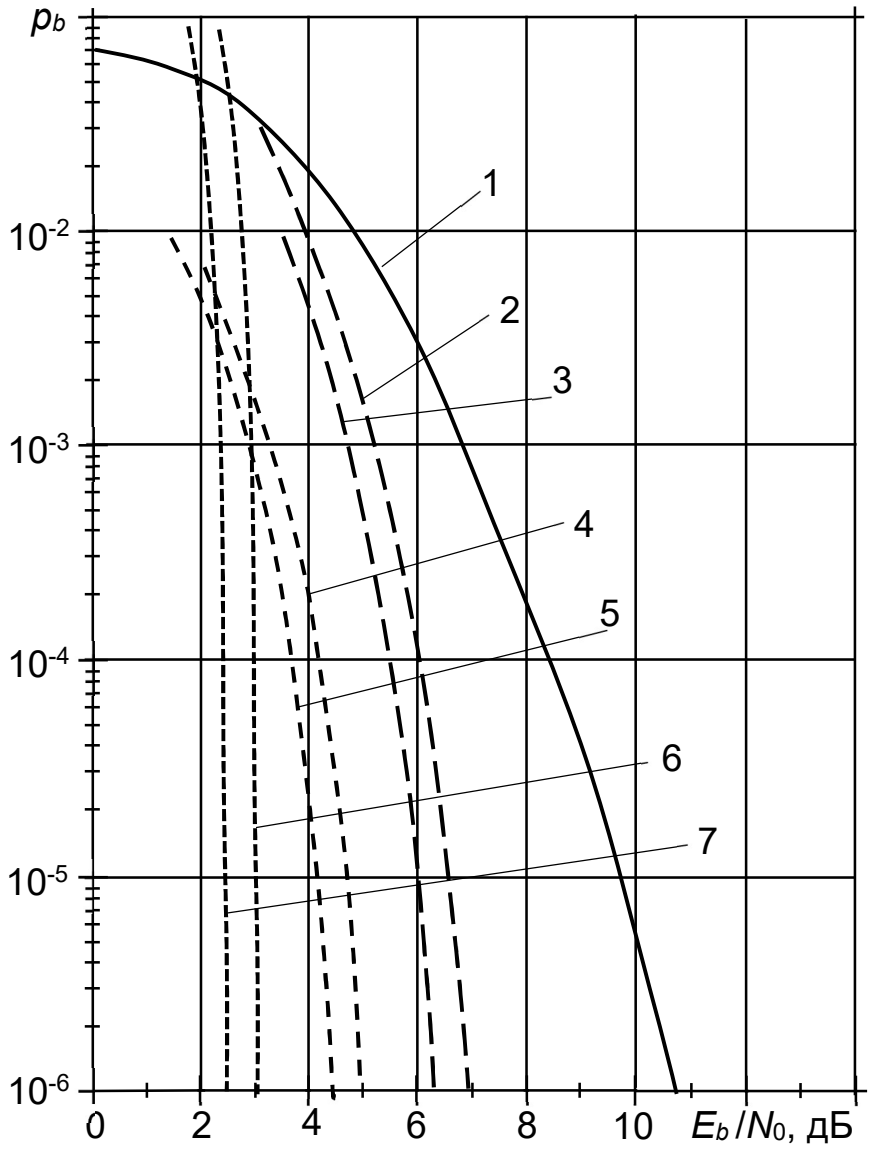


Рис. 3.79 Залежності імовірності появи бітової помилки p_b у цифрових сигналах з когерентною фазовою модуляцією від співвідношення потужності сигналу до потужності шуму у каналі зв'язку E_b/N_0 та від способів формування та декодування згорткового коду [33]: 1 – двійковий сигнал без кодування; 2 – двійковий код, $1/n = 1/2$, $K = 7$, алгоритм Вітербі; 3 – двійковий код, $1/n = 1/3$, $K = 7$, алгоритм Вітербі; 4 – багатопозиційний код, $1/n = 1/2$, $K = 7$, алгоритм Вітербі; 5 – багатопозиційний код, $1/n = 1/3$, $K = 7$, алгоритм Вітербі; 6 – двійковий код, $1/n = 1/2$, $K = 41$, послідовне декодування; 7 – двійковий код, $1/n = 1/3$, $K = 41$, послідовне декодування

3.8.7 Матричне подання згорткових кодів

Перед вивченням цього підрозділу необхідно повторити п'ятий розділ першого тому другої чистини посібника

У теорії кодування особлива увага приділяється згортковим кодам, призначеним для виявлення та виправлення пакетних помилок. Надамо відповідне визначення [63].

Визначення 3.23. Пакетною помилкою довжини t називається скінченна послідовність з t спотворених символів кодового слова, які йдуть підряд.

Виправлення пакетних помилок являє собою окреме складне завдання теорії згорткових кодів. Річ у тому, що у разі спотворення окремого символу повідомлення зазвичай таку помилку вдається легко визначити за метриками, оскільки як попередня, так і наступна послідовності є правильними. У разі наявності кількох окремих одиночних помилок ситуація не ускладняється, такі помилки можна легко визначити як за алгоритмом послідовного декодування, так і за алгоритмом Вітербі. Відповідні приклади були розглянуті та проаналізовані у підрозділах 3.8.5.1 та 3.8.6.3. Проте у разі виникнення пакетної помилки ситуація вкрай ускладняється. За умови такої помилки декодувальна система не може відразу визначити спотворений попередній символ за поточним станом системи, оскільки поточний символ також є спотвореним. Така невизначена ситуація буде продовжуватись, доки не закінчиться пакет помилок із довжиною t. Тоді, оскільки наступні кодові послідовності вже є правильними, декодер може проаналізувати пакет помилок із довжиною t та за схемою скінченного автомату через аналіз ґратки станів знайти шлях із найменшою метрикою.

Завдання пошуку пакетних помилок є вкрай важливим в теорії комунікаційних систем, оскільки саме такі помилки зазвичай виникають у каналах зв'язку у разі наявності довгочасної потужної завади [1 – 8, 33, 58]. У зв'язку з цим для виправлення пакетних помилок був розроблений

спеціальний клас удасконалених згорткових кодів [63]. Надамо відповідне визначення [63].

Визначення 3.24. Згортковий код, декодер якого може виправляти будь-які пакети помилок довжиною *t*, ізольовані один від одного, називається кодом, який здатний виправляти пакети помилок із довжиною *t*.

Зрозуміло, що будь-який згортковий код з параметрами $\left(K, \frac{1}{n}\right)$, який виправляє *t* незалежних помилок, здатний виправити також один пакет із *t* помилок. Нехай загальна кількість символів у коді складає *n* за умови кількості інформаційних символів *k*. Але, незважаючи на це, існують спеціальні способи формування згорткових кодів, призначених для виявлення та виправлення пакетів помилок.

Спочатку розглянемо матричне описання алгоритму формування згорткових кодів [63], яке загалом ґрунтується на поліноміальному описанні, розглянутому у підрозділі 3.8.4.1. У загальному випадку породжувальні поліноми з індексами *i* та *j* записуються у наступному вигляді [63]:

$$g_{i,j}(x) = \sum_i g_{i,j} x^i. \tag{3.313}$$

Для побудови відповідної породжувальної матриці необхідно впорядкувати поліноміальні коефіцієнти $g_{i,j}$ [63]. Тоді для кожного значення *l* коефіцієнти породжувального поліному можна записати у вигляді матриці $\mathbf{G}_l$ з розмірністю (k x l) [63]:

$$\mathbf{G}_l = g_{i,j,l}. \tag{3.314}$$

З урахуванням співвідношень (3.313), (3.314), породжувальна матриця для згорткового коду, який усічений до блокового коду із довжиною *n*, записується наступним чином [63]:

$$\mathbf{G}^{(n)} = \begin{bmatrix} \mathbf{G_0} & \mathbf{G_1} & \mathbf{G_2} & & \mathbf{G_m} \\ 0 & \mathbf{G_0} & \mathbf{G_1} & \cdots & \mathbf{G_{m-1}} \\ 0 & 0 & \mathbf{G_0} & & \mathbf{G_{m-2}} \\ & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \mathbf{G_0} \end{bmatrix}. \tag{3.315}$$

У загальному випадку породжувальною матрицею згортового коду є

матриця [63]:

$$\mathbf{G} = \begin{bmatrix} \mathbf{G_0} & \mathbf{G_1} & \mathbf{G_2} & & \mathbf{G_m} & 0 & 0 \\ 0 & \mathbf{G_0} & \mathbf{G_1} & \cdots & \mathbf{G_{m-1}} & 0 & 0 \\ 0 & 0 & \mathbf{G_0} & & \mathbf{G_{m-2}} & 0 & 0 \\ & \vdots & & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \mathbf{G_0} & 0 & 0 \end{bmatrix}. \qquad (3.316)$$

У формулі (3.316) породжувальна матриця може бути нескінченно продовженою донизу та праворуч.

Окремо у теорії кодування розглядаються породжувальні матриці для систематичних згорткових кодів, які, згідно із співвідношеннями (3.315) та (3.316), у загальному вигляді записуються наступним чином [63]:

$$\mathbf{G}^{(n)} = \begin{bmatrix} \mathbf{I} & \mathbf{P}_0 & \vdots 0 & \mathbf{P}_1 & \vdots 0 & \mathbf{P}_2 & \vdots 0 & \mathbf{P_m} \\ 0 & 0 & \vdots \mathbf{I} & \mathbf{P}_0 & \vdots \mathbf{I} & \mathbf{P}_1 & \vdots 0 & \mathbf{P_{m-1}} \\ 0 & 0 & \vdots 0 & 0 & \vdots \mathbf{I} & \mathbf{P}_0 & \vdots 0 & \mathbf{P_{m-2}} \\ \vdots & \vdots & \vdots\vdots & \vdots & \vdots\vdots & \vdots & \vdots\vdots & \vdots \\ 0 & 0 & \vdots 0 & 0 & \vdots 0 & 0 & \vdots \mathbf{I} & \mathbf{P}_0 \end{bmatrix}, \qquad (3.317)$$

$$\mathbf{G} = \begin{bmatrix} \mathbf{I} & \mathbf{P}_0 & \vdots\ 0 & \mathbf{P}_1 & \vdots\ 0 & \mathbf{P}_2 & \vdots\ 0 & \mathbf{P_m} & \cdots & \cdots \\ 0 & 0 & \vdots\ \mathbf{I} & \mathbf{P}_0 & \vdots\ \mathbf{I} & \mathbf{P}_1 & \vdots\ 0 & \mathbf{P_{m-1}} & \cdots & \cdots \\ 0 & 0 & \vdots\ 0 & 0 & \vdots\ \mathbf{I} & \mathbf{P}_0 & \vdots\ 0 & \mathbf{P_{m-2}} & \cdots & \cdots \\ \vdots & \vdots & \vdots\ \vdots & \vdots & \vdots\ \vdots & \vdots & \vdots\ \vdots & \vdots & \cdots & \cdots \\ 0 & 0 & \vdots\ 0 & 0 & \vdots\ 0 & 0 & \vdots\ \mathbf{I} & \mathbf{P}_0 & \cdots & \cdots \\ \cdots & \cdots & \cdots\ \cdots & \cdots & \cdots\ \cdots & \cdots & \cdots\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots\ \cdots & \cdots & \cdots\ \cdots & \cdots & \cdots\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}.$$

У співвідношеннях (3.317) кожний рядок отримується через зсув попереднього рядка на дів позиції праворуч, а невизначені матричні елементи ліворуч та праворуч від цього рядка дорівнюють 0.

Перевірочна матриця **H** для згорткового коду має задовольняти наступній умові [63]:

$$\mathbf{G}^{(ln_0)}\left(\mathbf{H}^{(ln_0)}\right)^{\mathrm{T}} = 0, l = 1,2,3, \ldots \qquad (3.318)$$

де $\mathbf{G}^{(ln_0)}$ та $\mathbf{H}^{(ln_0)}$ – підматриці, які стоять у лівих верхніх кутах матриць **G** та **H** та відповідають l кадрам згорткового коду. Згідно із співвідношенням (3.318) перевірочна матриця **H** може бути побудована безпосередньо за породжувальною матрицею **G**. Наприклад, як перевірочну можна обрати наступну матрицю [63]:

$$\mathbf{H} = \begin{bmatrix} \mathbf{P}_0^{\mathrm{T}} & -\mathbf{I} & \dots & \dots & \dots & \dots & & & \\ \mathbf{P}_1^{\mathrm{T}} & 0 & \mathbf{P}_0^{\mathrm{T}} & -\mathbf{I} & \dots & \dots & \dots & \dots & \dots \\ \mathbf{P}_2^{\mathrm{T}} & 0 & \mathbf{P}_1^{\mathrm{T}} & 0 & \mathbf{P}_0^{\mathrm{T}} & -\mathbf{I} & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{P}_m^{\mathrm{T}} & 0 & \mathbf{P}_{m-1}^{\mathrm{T}} & 0 & \mathbf{P}_{m-2}^{\mathrm{T}} & 0 & \dots & \mathbf{P}_0^{\mathrm{T}} & -\mathbf{I} \\ \dots & \dots & \mathbf{P}_m^{\mathrm{T}} & \dots & \mathbf{P}_{m-1}^{\mathrm{T}} & 0 & \dots & \dots & \dots \\ \dots & \dots & \dots & & \mathbf{P}_m^{\mathrm{T}} & 0 & \dots & \dots & \dots \end{bmatrix}. \qquad (3.319)$$

Слід відзначити, що значення **–I** у співвідношенні (3.319) стоїть лише за правилами матричних перетворень лінійної алгебри, які були розглянуті у четвертому розділі першого тому другої частини посібника. У двійковій арифметиці для одиничної матриці завжди виконується тотожність:

$$\mathbf{I} = -\mathbf{I}. \qquad (3.320)$$

Розглянемо кілька прикладів щодо формування породжувальних та перевірочних матриць для деяких згорткових кодів.

Приклад 3.61. Знайти породжувальну та перевірочну матриці згорткового коду з параметрами $n = 4$, $k = 2$, який описується елементарними породжувальними матрицями $\mathbf{P}_0 = 1$ та $\mathbf{P}_1 = 1$.

Враховуючи умову поставленого завдання, відповідно із співвідношеннями (3.317) та (3.319) маємо наступний результат:

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0\,1 & 0 & 0\,0 & 0 & 0 \\ 0 & 0 & 1\,1 & 0 & 1\,0 & 0 & 0 \\ 0 & 0 & 0\,0 & 1 & 1\,0 & 1 & 0 \\ \dots & \dots & \dots\dots & \dots & \dots\dots & \dots & \dots \\ \dots & \dots & \dots\dots & \dots & \dots\dots & \dots & \dots \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 & 1 & 0\,0 & 0 & 0\,0 & 0 & 0 \\ 0 & 0 & 1\,1 & 0 & 1\,0 & 0 & 0 \\ 0 & 0 & 0\,0 & 1 & 1\,0 & 1 & 0 \\ \dots & \dots & \dots\dots & \dots & \dots\dots & \dots & \dots \\ \dots & \dots & \dots\dots & \dots & \dots\dots & \dots & \dots \end{bmatrix}.$$

Приклад 3.62. Знайти породжувальну та перевірочну матриці згорткового коду з параметрами $n = 12$, $k = 4$, який описується елементарними породжувальними матрицями $\mathbf{P}_0 = [1, 1]$ та $\mathbf{P}_1 = [0, 1]$.

Враховуючи умову поставленого завдання, відповідно із співвідношеннями (3.317) та (3.319) маємо наступний результат:

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 1\,0 & 0 & 1\,0 & 1 & 0\,0 & 1 & 1\,0 & 0 & 0\,0 & 0 & 0\,0 & 0 & 0 \\ 0 & 0 & 0\,1 & 1 & 1\,0 & 0 & 1\,0 & 1 & 0\,0 & 1 & 1\,0 & 0 & 0\,0 & 0 & 0 \\ 0 & 0 & 0\,0 & 0 & 0\,1 & 1 & 1\,0 & 0 & 1\,0 & 1 & 0\,0 & 1 & 1\,0 & 0 & 0 \\ 0 & 0 & 0\,0 & 0 & 0\,0 & 0 & 0\,1 & 1 & 1\,0 & 0 & 1\,0 & 1 & 0\,0 & 1 & 1 \\ \dots & \dots & \dots\dots & \dots & \dots\dots & \dots & \dots\dots & \dots & \dots\dots & \dots & \dots\dots & \dots & \dots\dots & \dots & \dots \end{bmatrix},$$

$$\mathbf{H}=\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots
\end{bmatrix}.$$

Співвідношення (3.317), (3.319) використовуються на практиці для формування конструкцій згорткових кодів з використанням засобів комп'ютерної техніки. Тут головним завданням є формування конструкцій згорткових кодів, які дозволяють виявляти та виправляти пакетні помилки із довжиною t. Ефективних аналітичних методів формування таких кодових конструкцій сьогодні не існує.

Розглянемо конструкції деяких простих згорткових кодів, які дозволяють виявляти та виправляти одиночні помилки. Цей клас згорткових кодів називається кодами Вайнера – Еша. Вони аналогічні кодам Хеммінга, які розглядалися у підрозділі 2.4. Коди Вайнера – Еша формуються через породжувальну матрицю **G**, а декодер таких кодів – з використанням відповідної перевірочної матриці **H**.

Принцип формування кодів Вайнера – Еша полягає у тому, що для будь-якого натурального числа m можна сформувати конструкцію згорткового коду через перевірочну матрицю **H** із розмірністю $(2^m-1,\ 2^m-1-m)$, у якій всі 2^m-1 стовпчиків різняться та не є нульовими. Якщо для створення декодера згорткового коду використовувати елементарні породжувальні матриці $\mathbf{P}_0^{\mathrm{T}}, \mathbf{P}_1^{\mathrm{T}}, \dots, \mathbf{P}_m^{\mathrm{T}}$, елемент перевірочної матриці $\mathbf{H}^{2m(m+1)}$ буде мати наступний вигляд [63]:

$$\mathbf{H}^{2m(m+1)}=\begin{bmatrix}
\mathbf{P}_0^{\mathrm{T}} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{P}_1^{\mathrm{T}} & 0 & \mathbf{P}_0^{\mathrm{T}} & 1 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{P}_2^{\mathrm{T}} & 0 & \mathbf{P}_1^{\mathrm{T}} & 0 & \mathbf{P}_0^{\mathrm{T}} & 1 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbf{P}_m^{\mathrm{T}} & 0 & \mathbf{P}_{m-1}^{\mathrm{T}} & 0 & \mathbf{P}_{m-2}^{\mathrm{T}} & 0 & \dots & \mathbf{P}_0^{\mathrm{T}} & 1
\end{bmatrix}. \qquad (3.321)$$

Згідно із наведеним співвідношенням (3.321) повна матриця **H** для декодування послідовності коду Вайнера – Еша записується наступним чином [63]:

$$\mathbf{H} = \begin{bmatrix} \mathbf{P}_0^{\mathrm{T}} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{P}_1^{\mathrm{T}} & 0 & \mathbf{P}_0^{\mathrm{T}} & 1 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{P}_2^{\mathrm{T}} & 0 & \mathbf{P}_1^{\mathrm{T}} & 0 & \mathbf{P}_0^{\mathrm{T}} & 1 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{P}_m^{\mathrm{T}} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \mathbf{P}_m^{\mathrm{T}} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (3.322)$$

Наприклад, для коду Вайнера – Еша з параметром $m = 2$ співвідношення (3.320), (3.321) можна переписати нступним чином:

$$\mathbf{H}^{(12)} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix},$$

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}.$$

За таких умов породжувальна матриця коду Вайнера – Еша, усіченого до блоку із довжиною 12 бітів, складає [63]:

$$\mathbf{G}^{(12)} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Тоді у повному вигляді породжувальна матриця записується наступним чином [63]:

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}.$$

У наступному підрозділі будуть описані деякі прості цифрові схеми кодерів та декодерів згорткових кодів [63].

3.8.8 Прості структурні схеми кодерів та декодерів згорткових кодів

На рис. 3.80 показана структурна схема кодеру Вайнера – Еша, а на рис. 3.81 – відповідна схема декодера [63].

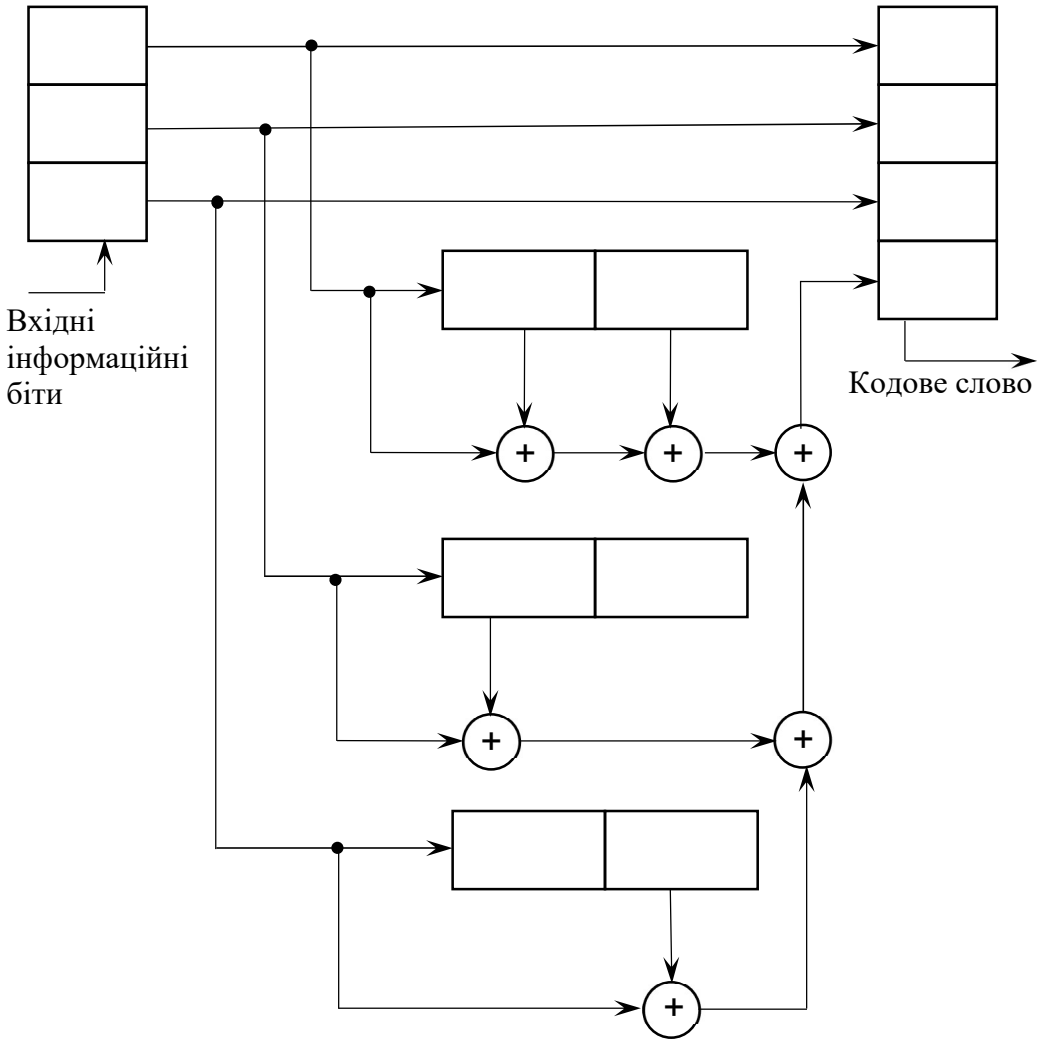


Рис. 3.80 Структурна схема кодеру Вайнера – Еша, який формує згортковий код з параметрами (12, 9)

Головним недоліком електронної схеми декодера, наведеної на рис. 3.81, є те, що з її використанням можна знаходити лише одиночні помилки і вона не призначена для пошуку пакетних помилок. Працює ця схема за алгоритмом послідовного декодування, який був описаний у підрозділі 3.8.6.1.

Реалізувати пошук багаторазових та пакетних помилок у згортковому коді можна або з використанням алгоритму послідовного декодування, або з використанням алгоритму Вітербі, описаного у підрозділі 3.8.6.3. Але як було

відмічено у підрозділі 3.8.6.4, для кодувальних та декодувальних схем із невеликою кількістю станів більш ефективним є алгоритм Вітербі. Інший спосіб пошуку помилок у згорткових кодах, який легко реалізувати на апаратному рівні, полягає у тому, що після знаходження кожної помилки встановлюється нульове значення регістру синдромам, але у цьому разі пошук пакетних помилок стає неможливим. Тобто, у разі апаратної реалізації такого способу, декодер буде шукати лише одиночні помилки [63].

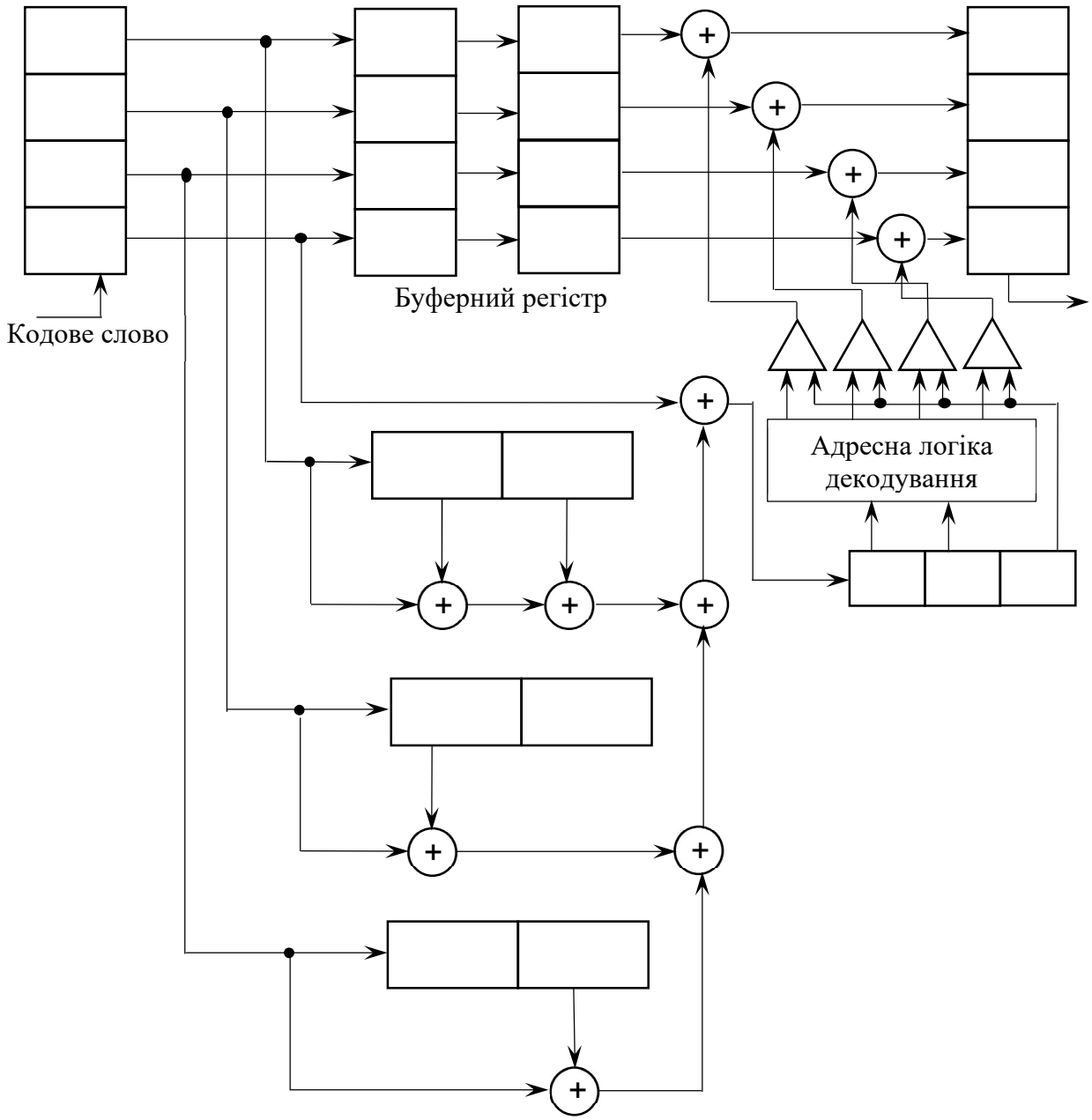


Рис. 3.81 Структурна схема декодера Вайнера – Еша, який розшифровує згортковий код з параметрами (12, 9)

Реалізація алгоритму декодування Вітербі на апаратному рівні полягає у тому, що в пам'яті декодера зберігається відповідна таблиця синдромів помилок. В процесі роботи декодера ці синдроми порівнюються із інформаційними повідомленнями, які надходять на вхід декодера, і саме таким чином помилки виправляються [63].

Розглянемо цифрову схему кодеру згорткового коду з параметрами (6, 3) та відповідного декодеру, який працює за алгоритмом синдромного декодування [63]. Відповідна цифрова схема кодеру наведена на рис. 3.82, а схема декодеру – на рис. 3.83.

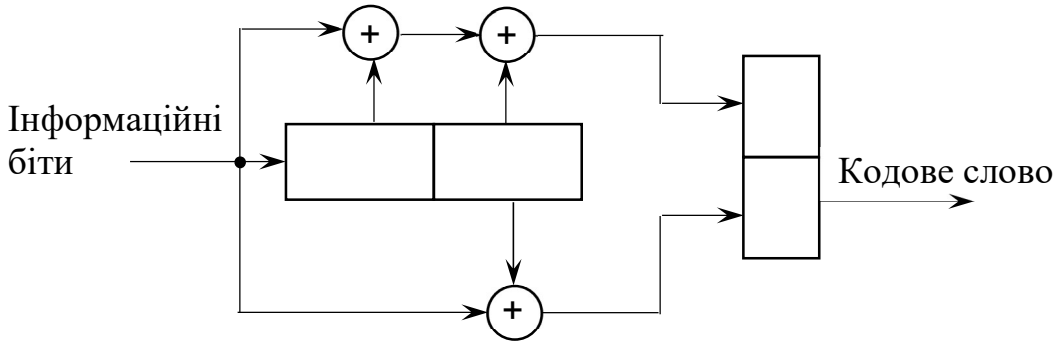


Рис. 3.82 Цифрова кодувальна схема для формування згорткового коду з параметрами (6, 3)

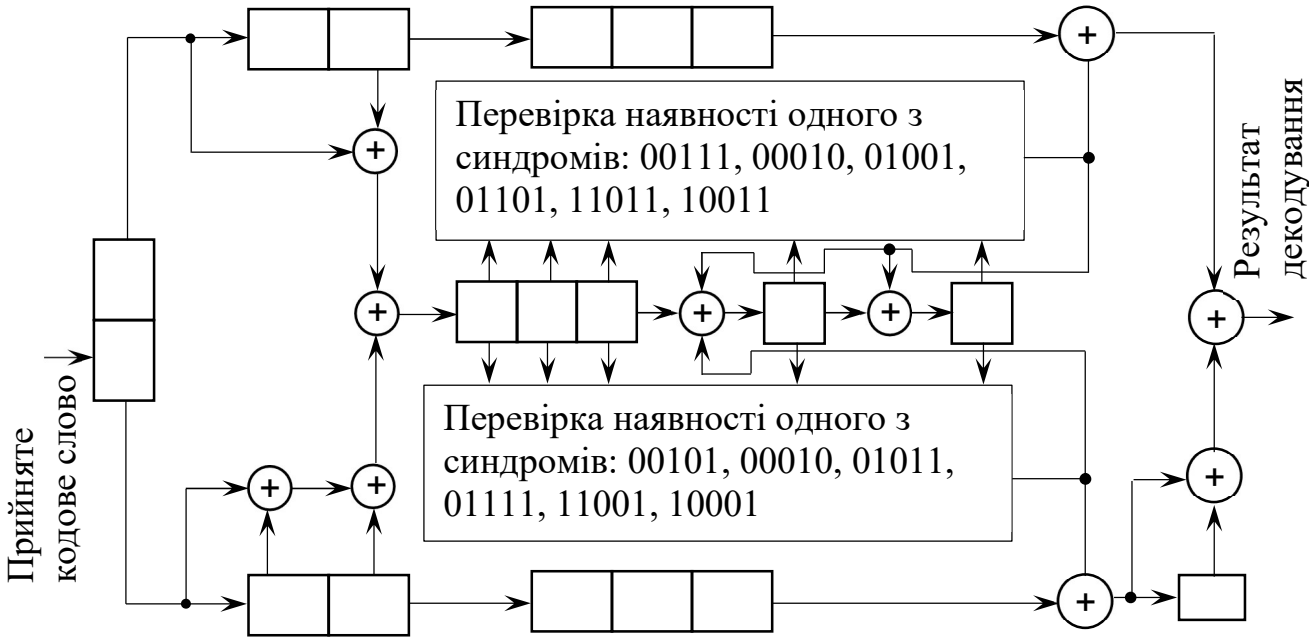


Рис. 3.83 Цифрова декодувальна схема для пошуку багаторазових помилок у згортковому коді з параметрами (6, 3)

Декодер, схема якого наведена на рис. 3.83, працює наступним чином. Він виправляє дві помилки в кожних шести бітах. За допомогою зворотного зв'язку, введеного в схему декодеру, внесок кожної помилки видаляється з регістру синдрому. Оскільки згортковий код (6, 3), який формується цифровою схемою, наведеною на рис. 3.82, не є систематичним, інформаційні символи поновлюються за виправленим кодовим словом. Для поновлення інформації використовується відповідне співвідношення дискретної математики [63]:

$1 = \text{НОД}[x^2 + x + 1, x^2 + 1] = x\,(x^2 + x + 1) + (x + 1)(x^2 + 1).$

Проте декодер, схема якого наведена на рис. 3.83, не є повним, оскільки існують синдроми помилок, які не входять до таблиці декодування. В принципі, він міг би виправляти й інші помилки, якщо включити в таблицю додаткові синдроми та, відповідно, збільшити довжину регістра синдромової пам'яті. Проте, з практичної точки зору, кращим способом поліпшення характеристик декодувальної системи зазвичай є обрання кодів з більшою довжиною елементарного блока. Недоліком такого способу є збільшення коефіцієнта надлишковості коду [63]. У будь-якому виапдку найкращим способом поліпшення характеристики декодеру згорткового коду є використання принципу максимальної правдоподібності та алгоритму Вітербі, описаного у підрозділі 3.8.6.4.

В таблиці 3.29 наведені деякі породжувальні поліноми для згорткових кодів, які отримані з використанням комп'ютерів та обчислювальних алгоритмів. У цій таблиці параметр *d* відповідає максимальній кодовій відстані між кодовими комбінаціями за Хеммінгом [63].

Таблиця 3.29 – Деякі породжувальні поліноми згорткових кодів [63]

Параметри коду (n, k, d)	Твірний поліном, поліноміальне подання	Твірний поліном, числове подання	Параметри коду (n, k, d)	Твірний поліном, поліноміальне подання	Твірний поліном, числове подання
(6, 3, 5)	$x^2 + 1$	101	(8, 4, 6)	$x^3 + x + 1$	1011
(10, 5, 7)	$x^4 + x^3 + 1$	11001	(12, 6, 8)	$x^5 + x^4 + x^2 + 1$	110101
(6, 3, 5)	$x^2 + x + 1$	111	(8, 4, 6)	$x^4 + x^3 + x^2 + 1$	1111
(10, 5, 7)	$x^5 + x^3 + x^2 + x + 1$	10111	(12, 6, 8)	$x^5 + x^3 + x^2 + x + 1$	101111

У наступному підрозділі розглядатимуться деякі ускладнені конструкції згорткових кодів, призначені для виявлення та виправлення пакетних помилок.

3.8.9 Ускладнені конструкції згорткових кодів для виправлення пакетних помилок та методи їх формування

Зрозуміло, що будь-який згортковий код із параметрами (n, k), який виправляє t помилок, буде виправляти також пакетні помилки, якщо довжина пакету є меншою за t. Тому завдання полягає в тому, щоб сформувати кодову конструкцію для виправлення пакетних помилок із більшою довжиною пакету, ніж t.

Згідно із загальними принципами теорії завадостійкого кодування такий згортковий код можна отримати з коду, який виправляє помилки кратності t, з використанням алгоритму перемежування [48, 63]. Припустимо, що існує згортковий код (n, k), який виправляє t помилок. Тоді, щоб отримати код з параметрами (jn, jk), здатний виправляти jt помилок, необхідно взяти j однакових кодерів із параметрами (n, k) та чередувати у створених повідомленнях вихідні символи [63]. Такий згортковий код не буде систематичним, але, незважаючи на це, його коректувальна здатність дозволятиме виправляти пакетні помилки з довжиною пакету jt.

Наведемо приклад формування згорткового коду через алгоритм перемежування. Розглянемо систематичний згортковий код (14, 7), який формується через наступні породжувальні поліноми:

$$g_1(x) = 1; \qquad g_2(x) = x^6 + x^5 + x^2 + 1. \tag{3.323}$$

Такий код має довжину пакету 14 та може виправляти на цій довжині дві будь-які помилки, зокрема пакетні. Тобто, для згорткового коду (14, 7) параметр t має значення 2. Проте, взявши чотири однакових коди з параметрами (14, 7), з використанням алгоритму перемежування отримуємо новий згортковий код з параметрами (56, 28), який має коректувальну здатність $t = 8$. Тоді породжувальні поліноми для створеного згорткового коду (56, 28), отримані через чотириразове множення поліномів (3.321), будуть мати наступний вигляд:

$$g_1(x) = 1; \qquad g_2(x) = x^{24} + x^{20} + x^8 + 1.$$

У навчальному посібнику [63] доведена теорема про те, що результатом

перемежування згорткових кодів завжди є згортковий код.

Таким чином, шляхом перемежування простих згорткових кодів, породжувальні поліноми для яких наведені у таблиці 3.29, можна отримувати згорткові коди із поліпшеною коректувальною здатністю, які будуть виправляти як випадкові, так і пакетні помилки. Проте, якщо згортковий код орієнтований на виправлення лише пакетних помилок, довжину кодової комбінації можна у значній мірі зменшити, якщо використовувати спеціальні кодові конструкції, призначені для виправлення саме таких помилок.

Одним з класів кодів, призначених для виправлення пакетних помилок, є коди Івадаре. Породжувальна матриця для коду Івадаре записується наступним чином [63]:

$$\mathbf{G}(x) = \begin{bmatrix} 1 & 0 & & 0 & 0 & g_1(x) \\ 0 & 1 & & 0 & 0 & g_2(x) \\ 0 & 0 & & & \cdots & 0 \quad \vdots \\ 0 & 0 & 0 & 1 & & g_{(n_0-1)}(x) \end{bmatrix}, \tag{3.324}$$

де для поліноміальних елементів матриці $\mathbf{G}(x)$ $g_{ni_0}(x)$ використовується наступне скорочене позначення:

$$g_{i_0}(x) = x^{(\lambda+1)(2n_0-i)+i-3} + x^{(\lambda+1)(n_0-i)-1},\ i = 1, 2, \ldots, n. \tag{3.325}$$

Згідно із співвідношенням (3.325), найбільший степінь, який дорівнює $(\lambda + 1)(2n_0 - 1) - 2$, має поліном $g_1(x)$. Для формування коду Івадаре, згідно зі співвідношеннями (3.324) та (3.325), використовуються довільні натуральні числа λ та n_0 [63]. Відповідно можна довести, що у разі формування коду Івадаре з використанням породжувальної матриці (3.324) з заданими значеннями натуральних чисел λ та n_0, він є згортковим кодом з наступними параметрами [63]:

$$((m + 1)n_0), (m + 1)(n_0 - 1)), \quad m = (\lambda + 1)(2n_0 - 1) - 2. \tag{3.326}$$

Також можна довести, що згортковий код Івадаре виправляє пакетні помилки з довжиною λn_0 пакету [63]. Також можна показати, що перевіорчна матриця коду Івадаре, з урахуванням співвідношення (3.324), має наступний вигляд [63]:

$$\mathbf{H}(x) = [g_1(x) \quad g_2(x) \quad \cdots \quad g_{(n_0-1)}(x) \quad 1]. \tag{3.327}$$

З використанням відповідних теорем дискретної математики можна також довести, що у разі виконання співвідношення $n_0 - k_0 = 1$ існує лише один відповідний синдромний поліном, призначений для пошуку пакетних помилок [63]:

$$s(x) = \sum_{i=0}^{n_0-1} g_i(x)e_i(x) + e_{n_0}(x) = e_{n_0}(x) + \\ + \sum_{i=0}^{n_0-1} \left[x^{(\lambda+1)(n_0-i)-1} + x^{(\lambda+1)(2n_0-i)+i-3}\right]e_i(x). \quad (3.328)$$

Декодер коду Івадаре використовує синдромний поліном (3.328) для виявлення та виправлення пакетних помилок. Наприклад, припустимо, що помилки починаються в першому кадрі та пакет містить λn_0 бітів. Але можливою є також інша ситуація, а саме, коли пакет починається не з першого кадру та, відповідно, захоплює кадр з номером $(\lambda + 1)$ [63]. Щоб остаточно довести, що код Івадаре може виправляти такі пакетні помилки, необхідно обґрунтувати дві наступні його властивості [63].

Властивість 3.16. Декодер коду Івадаре з використанням синдромного поліному (3.328) може поновити інформаційне повідомлення із пакетною помилкою, якщо довжина пакету помилок не перевищує λn_0 бітів.

Властивість 3.17. Опрацювання з використанням синдромного поліному (3.328) пакету помилок, який може починатися у будь-якому кадрі інформаційного повідомлення, за будь-якої умови не може привести до виникнення помилки в роботі декодера коду Івадаре.

Оскільки за свою структурою всі коди Івадаре є однаковими, найпростіше за все пояснити виконання властивостей 3.16 та 3.17 на конкретному прикладі дешифрування коду із заданою коректувальною здатністю. Розглянемо такий приклад [63].

Приклад 3.63. Проаналізувати роботу декодера коду Івадаре з параметрами (72, 48), $n_0 = 3$ та $\lambda = 4$.

Спочатку, згідно із другим співвідношенням системи рівнянь (3.326), знайдемо параметр m:

$m = (\lambda + 1)(2n_0 - 1) - 2 = (4 + 1)(2 \cdot 3 - 1) - 2 = 5(6 - 1) - 2 = 5 \cdot 5 - 2 = 23$,

тобто, $m + 1 = 24$.

Структурна схема кодеру коду Івадаре з параметрами (72, 48) представлена на рис. 3.84 [63].

Для подальшого аналізу коректувальної здатності коду Івадаре позначимо через $c_1(x)$ та $c_2(x)$ поліноми, що описують інформаційні біти, а через $c_3(x)$ – відповідний поліном, який описує контрольні біти. У такому разі для кожного з цих поліномів можна сформувати поліноми відповідні поліноми для генерації помилок, які позначимо як $e_1(x)$, $e_2(x)$ та $e_3(x)$. За таких умов, згідно із співвідношенням (3.328) можна записати наступний синдромний поліном [63]:

$$s(x) = e_3(x) + (x^4 + x^{10})e_2(x) + (x^9 + x^{23})e_1(x). \qquad (3.329)$$

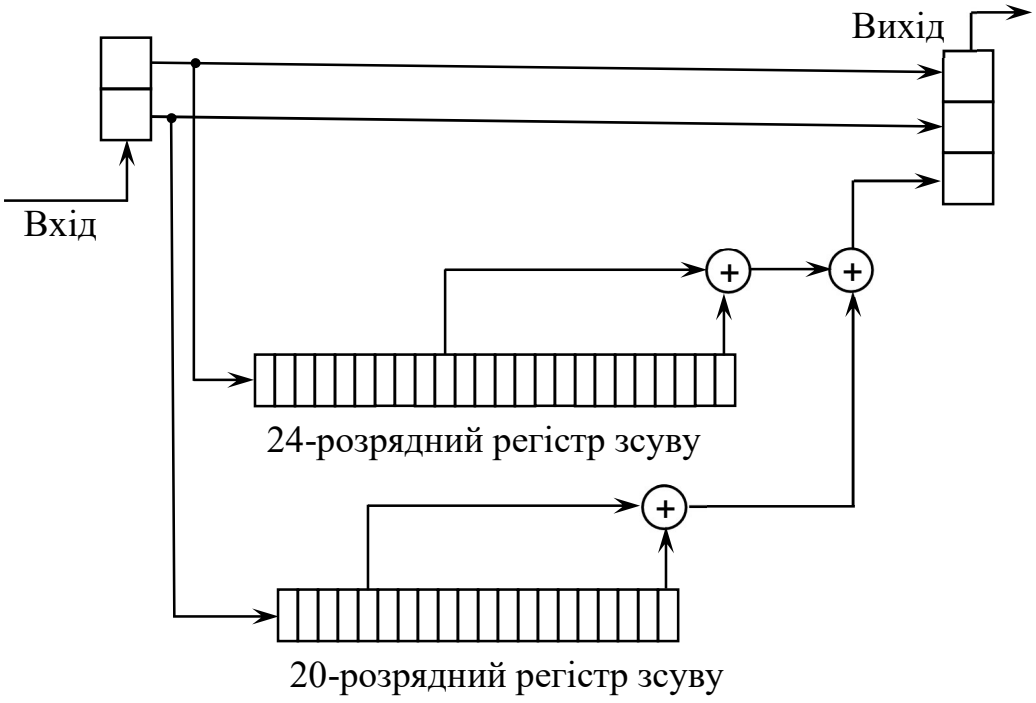


Рис. 3.84 Узагальнена структурна схема цифрового електронного пристрою для формування коду Івадаре з параметрами (72, 48)

У співвідношенні (3.329) значення векторів помилок $e_1(x)$, $e_2(x)$ та $e_3(x)$ є наступними [63]:

$$e_3(x) = e_{30} + e_{31}x + e_{32}x^2 + e_{33}x^3,\ e_2(x) = e_{20} + e_{21}x + e_{22}x^2 + e_{23}x^3 + e_{24}x^4,$$

$$e_1(x) = e_{10} + e_{11}x + e_{12}x^2 + e_{13}x^3 + e_{14}x^4. \qquad (3.330)$$

З наведених співвідношень (3.330) видно, що якщо пакет помилок починається з першого кадру, коефіцієнт e_{34} дорівнює 0. З іншого боку, якщо коефіцієнт e_{14} є ненульовим, тоді $e_{10} = 0$ [63]. Ці властивості коду Івадаре (72, 48) безпосередньо впливають з того, що довжина пакету помилок не перевищує значення 12. Наведемо коефіцієнти $s(x)$ для трьох випадків, коли пакет помилок починається відповідно з векторів e_{10}, e_{20} та e_{30} [63].

Для випадку, коли вектор помилок починається з e_{10}.

Відгук: $e_1(x) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix}$; $e_2(x) = \begin{bmatrix} 2 & 2 & 2 & 2 \\ 3 & 2 & 1 & 0 \end{bmatrix}$;

$$e_3(x) = [0 \quad 0 \quad 0 \quad 0 \quad 0].$$

$$e_1(x) = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 3 & 2 & 1 & 0 \end{bmatrix}; \; e_2(x) = \begin{bmatrix} 0 & 2 & 2 & 2 & 2 \\ 0 & 3 & 2 & 1 & 0 \end{bmatrix};$$

$$e_3(x) = \begin{bmatrix} 3 & 3 & 3 & 3 \\ 3 & 2 & 1 & 0 \end{bmatrix}.$$

Для випадку, коли вектор помилок починається з e_{20}.

Відгук: $e_1(x) = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 4 & 3 & 2 & 1 & 0 \end{bmatrix}$; $e_2(x) = \begin{bmatrix} 2 & 2 & 2 & 2 \\ 3 & 2 & 1 & 0 \end{bmatrix}$;

$$e_3(x) = [0 \quad 0 \quad 0 \quad 0 \quad 0].$$

$$e_1(x) = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 4 & 3 & 2 & 1 & 0 \end{bmatrix}; \; e_2(x) = \begin{bmatrix} 0 & 2 & 2 & 2 & 2 \\ 0 & 3 & 2 & 1 & 0 \end{bmatrix};$$

$$e_3(x) = \begin{bmatrix} 3 & 3 & 3 & 3 \\ 3 & 2 & 1 & 0 \end{bmatrix}.$$

Для випадку, коли вектор помилок починається з e_{30}.

Відгук: $e_1(x) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 4 & 3 & 2 & 1 \end{bmatrix}$; $e_2(x) = \begin{bmatrix} 2 & 2 & 2 & 2 & 0 \\ 4 & 3 & 2 & 1 & 0 \end{bmatrix}$;

$$e_3(x) = [0 \quad 0 \quad 0 \quad 0 \quad 0].$$

$$e_1(x) = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 \\ 4 & 3 & 2 & 1 & 0 \end{bmatrix}; \; e_2(x) = \begin{bmatrix} 2 & 2 & 2 & 2 & 0 \\ 4 & 3 & 2 & 1 & 0 \end{bmatrix};$$

$$e_3(x) = \begin{bmatrix} 3 & 3 & 3 & 3 \\ 3 & 2 & 1 & 0 \end{bmatrix}.$$

Слід відзначити, що для створення синдромного поліному коду Івадаре (3.329) витримані наступні правила формування таких поліномів [63].

1. На кожний біт синдрому помилки впливає лише один помилковий біт.

2. Якщо спочатку поліноми є перемежаними та розташовані в прямому порядку $e_1(x)$, $e_2(x)$, $e_3(x)$, потім над їх зображенням виконується зворотне

перемежування та вони розташовуються в зворотному порядку, $e_3(x)$, $e_2(x)$, $e_1(x)$.

У цьому посібнику вже неодноразово відмічалось, що порядок слідування розрядів у кодах є одним із найважливіших принципів їхнього формування, якого слід дотримуватись та завжди його ретельно перевіряти. Саме неправильний порядок розташування розрядів у коді може стати головною причиною виникнення помилок у недосвідчених математиків, програмістів та інженерів, які починають займатися теорією кодування та формуванням кодів для розв'язування практичних завдань проектування сучасної цифрової електронної апаратури.

Кожний біт поліному $e_2(x)$ діє на синдром помилки двічі, після його першої появи надходить відлуння на цю подію, і затримка між першою появою та відлунням складає 15 бітів. Аналогічно двічі діє на синдром помилки поліном $e_1(x)$, тут затримка між першою появою та відлунням складає 14 бітів. Оскільки вектори e_{24} та e_{10} не можуть одночасно приймати ненульове значення, ці локатори помилок ніколи не перетинаються один з одним. На відміну від поліномів $e_1(x)$ та $e_2(x)$ кожен біт поліному $e_3(x)$ діє на синдром помилки лише один раз. З появою цього синдрому надходять два нулі, один з затримкою 14 бітів, а другий – з затримкою 15 бітів. Саме ці нулі і відрізняють синдромний поліном $e_3(x)$.

Якщо пакет помилок починається не в першому кадрі коду Івадаре, а в кадрі з номером l, вектор синдрому помилок автоматично зсувається на l позицій праворуч та починається з послідовності з l нулів. Схема декодера та процедура декодування побудовані таким чином, що l нулів на початку вектору синдрому помилок свідчать про те, що в перших l кадрах помилки відсутні.

Узагальнена структурна схема декодера коду Івадаре наведена на рис. 3.85. Розглянемо головні особливості роботи цієї схеми.

Після того, як на вхід декодеру надійшли 20 кадрів, у регістрі синдрому вже завантажені 20 бітів, які характеризують наявність помилки. Перший біт синдрому помилки s_0 розташований праворуч. Аналогічно, перший прийнятий біт поліному $v_2(x)$ розташований у крайньому правому розряді відповідного регістра. Протягом ще чотирьох наступних тактів перший прийнятий біт послідовності $v_1(x)$ не досягає крайнього правого розряду відповідного регістра. Крім того, перші чотири біти регістра синдрому залежать лише від помилок перевірочних бітів та їх слід аналізувати окремо [63].

Тобто, будемо проводити аналіз роботи декодеру коду Івадаре, схема якого наведена на рис. 3.85, починаючи з 24 такту. Спочатку декодер виправляє другий біт кожного кадру з пакету помилок, і лише після цього повертається автоматично до першого біту цього кадру. У зв'язку з цим до схеми декодеру введено чотирьохрозрядний регістр зсуву. В цьому регістрі зберігається послідовність коректувального поліному $c_2(x)$ після завершення операції виправлення другого біту кадру. Третій біт кожного кадру є контрольним бітом, тому додаткова перевірка цього біту є зайвою [63].

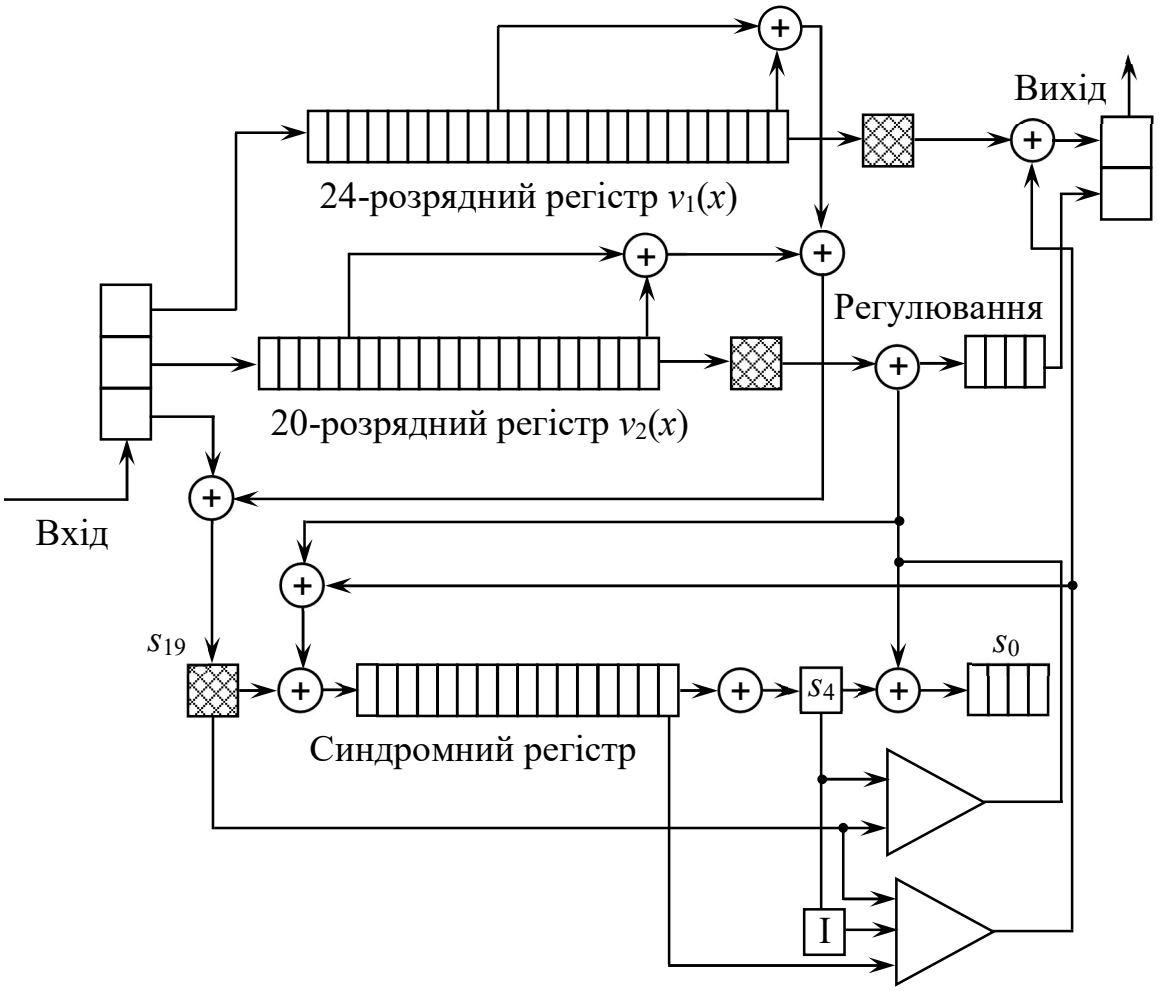


Рис. 3.85 Узагальнена структурна схема цифрового електронного пристрою для декодування коду Івадаре з параметрами (72, 48)

З метою виправлення помилки у послідовності $v_2(x)$ декодер перевіряє компоненти s_4 та s_{19} синдрому помилок. Відповідні регістри, в яких зберігаються ці бітові послідовності, позначені на схемі, наведеній на

рис. 3.85. Якщо обидві ці компоненти синдрому помилок дорівнюють 1, тоді правий біт послідовності $v_2(x)$ вважається помилковим та автоматично виправляється у потоці даних. Одночасно вносяться відповідні правки до регістру синдрому помилок, для чого використовуються показані на схемі ланцюги зворотного зв'язку. Пакет помилок $e_3(x)$ починається пізніше та розташований в іншому кадрі потоку даних. За такої умови в першому кадрі він не впливає на алгоритм роботи схеми декодування та відгук схеми декодера на цей вектор дорівнює нулю. Паралельно з аналізом бітів s_4 та s_{19} вектору синдрому помилок та виправленням вхідної послідовності $v_2(x)$, декодер шукає помилки у векторі $v_1(x)$ за компонентами синдрому помилок s_5 та s_{19}, але цей аналіз здійснюється на чотири кадри пізніше. Якщо компоненти s_5 та s_{19} дорівнюють 1, а попередня перевірка показала, що $s_4 = 0$, тоді правий біт вектору $v_1(x)$ вважається помилковим та виправляється.

Важливою умовою коректної роботи декодувальних схем коду Івадаре є те, що після появи пакету помилок із заданою максимальною довжиною l подальша прийнята інформація протягом певного часу має бути безпомилковою [63]. Лише у разі виконання цієї умови декодер може виправити всі помилки, наявні у пакеті, за сформованим вектором синдрому помилок. Для схеми, наведеної на рис. 3.85, безпомилковими повинні бути 24 кадри, які йдуть після помилкових, а кількість бітових помилок у помилкових кадрах не має перевищувати 12, що відповідає чотирьом кадрам.

3.8.10 Алгоритм Фано та його апаратна реалізація на логічних схемах та регістрах зсуву

Розглянемо у цьому підрозділі один із найбільш ефективних варіантів практичної реалізації алгоритму послідовного декодування згорткового коду. Цей алгоритм у загальному вигляді та із наочними прикладами був описаний у підрозділі 3.8.6.1. Реалізація алгоритму послідовного декодування, яка розглядатиметься у цьому підрозділі, називається алгоритмом Фано. Головна перевага цього алгоритму полягає в тому, що його можна легко реалізувати у

вигляді простої цифрової електронної схеми, головними елементами якої є програмована логіка, схеми «Виключного АБО» та регістри зсуву [63].

Для реалізації алгоритму послідовного декодування на апаратному рівні насамперед необхідно знати імовірність p появи хибного символу в каналі цифрової системи зв'язку, або, принаймні, верхнє граничне значення цього важливого параметру [1]. Тому загалом, з теоретичної точки зору, алгоритм Фано ґрунтується на простій гіпотезі про те, що у випадку, коли шлях, за яким йде розшифрування згорткового коду у декодувальному пристрої, є правильним, найбільш імовірна кількість помилок у l кадрах коду складає приблизно pn_0l, де n_0 – кількість символів у кадрі [63].

Відповідно до сформульованої гіпотези розглянемо параметр p', діапазон значень якого лежить в межах

$$p < p' < \frac{1}{2}. \tag{3.331}$$

Зазвичай значення параметру p' обирають в числовому діапазоні, який задається співвідношенням (3.331). Для визначення числового значення цього параметру можуть бути використані відомі методи стохастичного моделювання поширення електромагнітного сигналу в каналі зв'язку приймально-передавальної системи. Відповідні способи моделювання були описані у першій частині посібника [1].

За умови відомого значення p' можна знайти критичну відстань згорткового коду $t(l)$. У разі зменшення критичної відстані вважається, що обраний за ґратковою діаграмою шлях може бути хибним, тобто, необхідно перевірити інші можливі шляхи. Значення $t(l)$ обчислюється після розшифрування кожного кадру коду з використанням аналітичного співвідношення [63]:

$$t(l) = p'n_0l - d(l), \tag{3.332}$$

де $d(l)$ – кодова відстань за Хеммінгом між прийнятою послідовністю символів та очікуваною послідовністю, яка має бути на поточний момент згідно із структурою ґратки.

Слід відзначити, що, згідно з теорією кодування, у разі пересування

вздовж ґратки за правильним шляхом завжди виконується співвідношення

$$d(l) \approx p' n_0 l, \tag{3.333}$$

тобто, значення $t(l)$, обчислене з використанням співвідношення (3.332), завжди є позитивним та збільшується з кожним кадром в процесі аналізу отриманих закодованих повідомлень [63].

Згідно з наведеними вище теоретичними міркуваннями алгоритм Фано можна описати наступним чином. Якщо динаміка зростання значень $t(l)$ відповідає співвідношенням (3.332) та (3.333), вважається, що декодер опрацьовує кодову послідовність правильно та не треба вносити ніяких змін в алгоритм його роботи. Проте якщо раптово значення $t(l)$ в процесі послідовної обробки кадрів починають зменшуватись, вважається можливим, що в одному з попередніх вузлів була обрана помилкова гілка ґраткової діаграми та, в зв'язку з цим, на цьому етапі розшифрування кодової послідовності необхідно перевірити можливі альтернативні шляхи. А надалі можливими є наступні два варіанти.

1. Кодова відстань за метрикою альтернативного шляху є меншою і тому саме цей шлях вважається правильним.

2. Кодова відстань за метрикою альтернативного шляху є більшою, а тоді необхідно знову повернутися до шляху, який був обраний декодером раніше.

Тобто, згідно з алгоритмом Фано, шлях, на якому починають зменшуватись значення $t(l)$, вважається не остаточно хибним, а можливо хибним. Він, за будь яких умов, не відкидається з розгляду, просто послідовно перевіряються можливі альтернативні варіанти шляху та обчислюються метрики відповідних гілок ґраткової діаграми згортового коду. Остаточно правильним вважається той шлях, для якого значення $t(l)$ є максимальним та відповідає співвідношенню (3.333).

Для того, щоб вирішити, коли саме зменшення значення $t(l)$ досягає критичної величини, для декодувальної схеми встановлюється поточний поріг T, який завжди є кратним значенню приросту порогу Δ [63]. Коли декодер

рухається вперед, тобто, від кореня до вершини ґраткової діаграми, поріг T залишається меншим за значення $t(l)$ та кратним Δ, проте значення цієї величини, з урахуванням умов (3.331) та (3.333), є максимально можливим. Крім цього, враховуючи те, що значення T, згідно із співвідношенням (3.332), є квантованими, завжди припускається можливість невеликого зменшення значення $t(l)$, але без його виходу за межі порога [63].

Ще однією вимогою стосовно реалізації алгоритму Фано є те, що всі q^{k_0} ребр ґраткової діаграми, які виходять з кожного з вузлів, мають бути пронумеровані відповідно до певних правил впорядкування [63]. Проте запам'ятовувати індекси кожного ребра немає потреби. Цілком достатньо знати єдине правило, за яким, у разі повернення декодера до відповідного вузла через ребро з номером j, можна заново впорядкувати ребра. Це дозволяє знайти як ребро j, так і ребро $j + 1$, яке розташоване поряд з ребром j за структурою ґраткової діаграми згортового коду. Найбільш загальним з таких правил є правило мінімуму кодової відстані [63]. Тобто, зазвичай ребра впорядковуються згідно з відстанню Хеммінга між кадром прийнятого слова та прогнозованою кодовою комбінацією. У підрозділах 3.8.6.1 та 3.8.6.3 було показано, що у разі використання цього правила на відповідних етапах декодування можуть виникати невизначеності, які завжди слід враховувати під час практичної реалізації алгоритму послідовного декодування, але правила обробки таких невизначеностей є довільними. Існують також інші способи впорядкування ребр ґраткової діаграми згортового коду для реалізації алгоритму Фано. Наприклад, це можна зробити лексикографічним способом згідно з послідовністю символів, яка відповідає певному ребру. Припустимо, ребру можна надати ім'я «1001». Така систематизація ребр, з одного боку, призводить до ускладнення їхнього пошуку під час руху вздовж ґратки у прямому та зворотному напрямках, але за такої умови відпадає необхідність повторного розрахунку номерів ребр у разі повернення до відповідного вузла. Крім цього, систематизація ребр за іменами є більш зручною та зрозумілою для інженерів-проектувальників, які розробляють кодувальне електронне

обладнання, а цей важливий економічний чинник сприяє прискоренню розробки кодувального електронного обладнання, а також спрощує його ремонт та налагодження [8]. У загальному випадку правило систематизації ребр визначається конструкцією конкретного декодеру [63]. Проте для подальшого розуміння описання алгоритму Фано будемо вважати першим те ребро, яке є найближчим до поточного за відстанню Хеммінга. Саме такий принцип систематизації ребр був розглянутий у підрозділі 3.8.6.1, а у разі виникнення невизначеності ребро обиралося довільно випадковим чином.

Зрозуміло, що поки значення $t(l)$ є більшим за поріг T, декодер аналізує прийняті кодові послідовності та з кожним кадром підвищує значення порогу згідно із співвідношенням (3.333). А якщо значення $t(l)$ зменшується та стає меншим за порогове значення, декодер намагається знайти альтернативне ребро ґратки, для якого $t(l)$ перевищує порогове значення. Якщо почався процес руху декодера у зворотному напрямку, від поточного вузла до кореня ґраткового дерева, тоді проводиться перевірка всіх альтернативних ребр, з метою знайти таке з них, метрика якого є вищою за порогове значення. Після цього процес декодування продовжується далі в прямому напрямку, від кореня до вершини ґраткової діаграми, але в результаті аналізу, проведеного у зворотному напрямку, значення порогу T стає меншим.

Важливим правилом для забезпечення коректної роботи алгоритму Фано є те, що декодер може повернутися до вузла ґратки, який вже аналізувався раніше, лише у разі зменшення значення порогу T [63]. В наслідок цього кожен вузол ґратки за будь-якої умови може аналізуватися лише певну кількість разів. Це дозволяє уникнути можливості створення вічних циклів в процесі аналізу помилкових кодових комбінацій. Можливим є лише хибне розшифрування кодової комбінації у разі наявності великої кількості помилок.

Сформулюємо ще два важливих правила, які покладені в основу алгоритму Фано [63].

1. За умови, якщо декодер не може знайти альтернативний шлях, він повертається до вузла, де було встановлене відповідне значення порогу, та

зменшує його. Тобто, якщо пошук альтернативного шляху від вузла j у зворотному напрямку не дав результатів та декодер повернувся до цього вузла у прямому напрямку, поточна величина порогу T_j зменшується на фіксовану величину Δ [63]:

$$T_j = T_j - \Delta. \tag{3.334}$$

2. Декодер підвищує значення порогу лише у тому випадку, коли проводиться аналіз нового вузла. Тобто, у разі проходження нового вузла поріг T_j обчислюється ітераційно за наступним співвідношенням [63]:

$$T_j = T_{j-1} + \Delta. \tag{3.335}$$

Максимальне значення порогу для кадру з номером l обчислюється за співвідношенням дискретної математики [48, 63]:

$$T = \Delta \left[\frac{t_l}{\Delta}\right]. \tag{3.336}$$

Після того, як поріг зменшений на величину Δ згідно із співвідношенням (3.334), декодер аналізує наступні ребра в прямому напрямку. В процесі цього аналізу значення порогу T не змінюється, доки воно не перевищує критичну величину. Але коли декодер доходить до вузла l, в якому виконуються нерівності [63]:

$$t(l-1) < T_{l-1} + \Delta,\ t(l-1) \geq T, \tag{3.337}$$

необхідно збільшити значення порогу T_l згідно із співвідношенням (3.335).

Тобто, саме виконання умов (3.337) свідчить про те, що відповідний вузол ґраткової діаграми ще не аналізувався. Тому під час руху у зворотному напрямку та повернення за структурою дерева відпадає необхідність запам’ятовувати всі проаналізовані вузли, оскільки пошук нового вузла здійснюється автоматично за співвідношеннями (3.337).

Згідно з наведеними теоретичними відомостями, всі кроки алгоритму Фано насамперед визначаються двома фіксованими параметрами, p' та Δ. Ці параметри зазвичай визначаються з використанням комп’ютерних методів стохастичного моделювання поширення електромагнітного сигналу у каналі зв’язку [1]. Проте на практиці завжди необхідно зменшувати значення $t(l)$ та T у разі, якщо вони перевищують задану критичну величину. Слід відзначити, що віднімання від значень $t(l)$ та T невеликої величини, кратної Δ, загалом не впливає на обчислювальні особливості алгоритму Фано та на його збіжність.

З використанням відомих співвідношень дискретної математики, розглянутих у другій частині посібника [48 – 50], можна довести відповідні теореми теорії згорткових кодів [63].

У разі практичного використання алгоритму Фано слід також обирати відповідну величину вікна декодування b, яка повинна бути в кілька разів більшою за величину кодового блока. Параметр b вказує на останній кадр, який зберігається в буфері декодера та до якого він може повернутися у разі необхідності опрацювання кадрів у зворотному напрямку [63]. У противному випадку декодер може послатися на дуже старий кадр, який вже був раніше вивантажений з буфера, а таке посилання розглядається як помилка алгоритму Фано.

Узагальнена блок-схема алгоритму Фано, на якій відображена важлива роль внутрішніх параметрів декодеру T, Δ та b, наведена на рис. 3.86. Слід відзначити, що параметри Δ та b є незмінними. На цій блок-схемі також введений додатковий внутрішній параметр M, який характеризує напрямок аналізу ґраткової діаграми згорткового коду. За умови $M = 1$ здійснюється рух у прямому напрямку, у разі $M = -1$ – у зворотному, а за умови $M = 0$ проводиться аналіз поточного вузла ґратки з меншим значенням порогу T_j, яке обчислюється за співвідношенням (3.334).

Можливою технічною помилкою роботи схеми декодеру є перевантаження вхідного буферу, і саме можливість виникнення цієї помилка накладає суттєві обмеження щодо практичного використання алгоритму Фано в сучасній електронній апаратурі [63]. Слід відзначити, що збільшення розміру буферу не впливає суттєво на зменшення імовірності його переповнення. Це твердження можна довести з використанням відомих математичних методів теорії черг, які розглядалися у другому томі другої частини посібника [49]. Особливості використання параметру b та проблема переповнення буфера, як загальний недолік алгоритму послідовного декодування, розглядалися у підрозділах 3.8.6.1 та 3.8.6.4. Порівняльний аналіз алгоритмів послідовного декодування та Вітербі для згорткових кодів з різними параметрами був проведений у підрозділі 3.8.6.4.

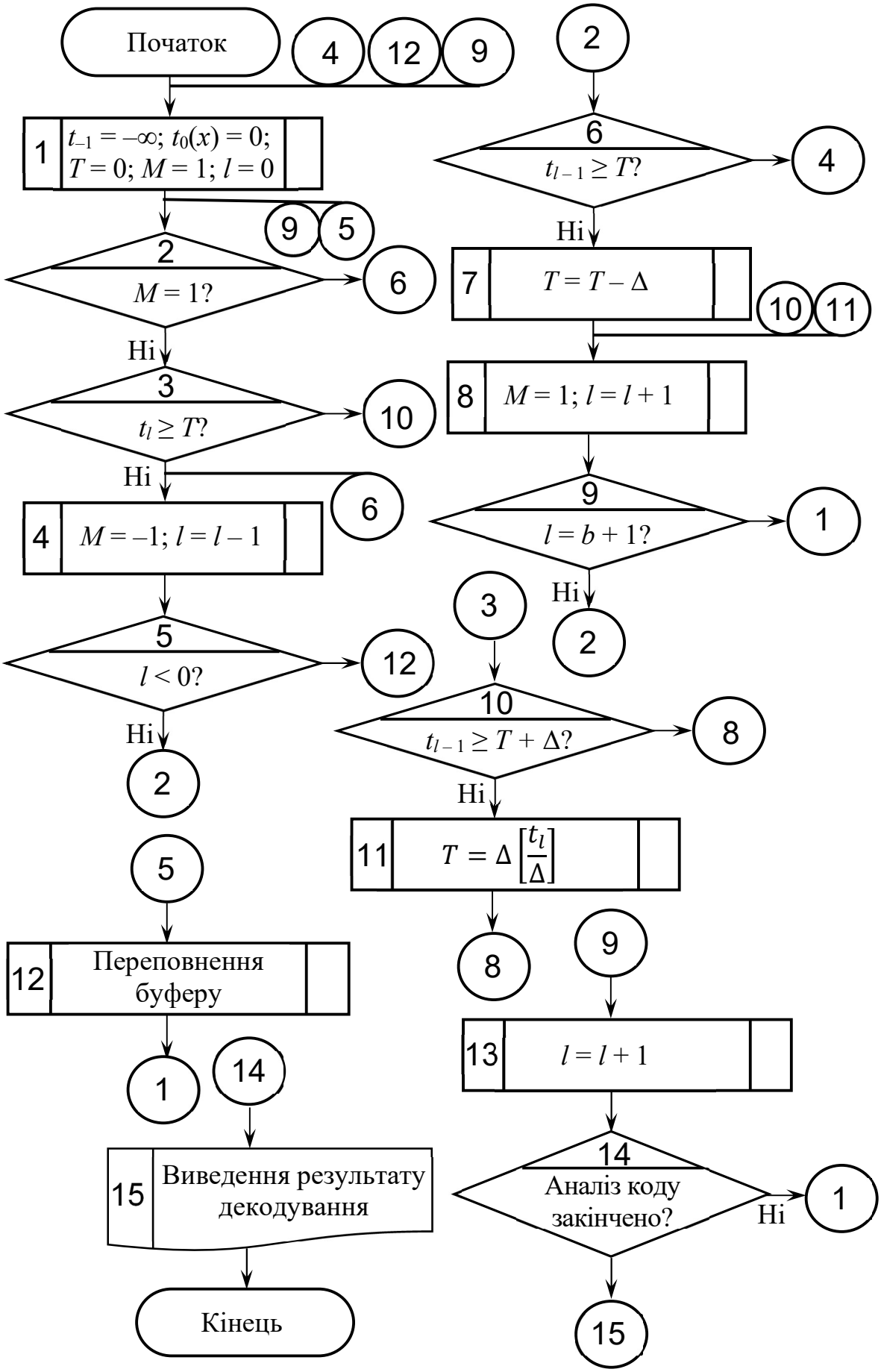


Рис. 3.86 Узагальнена блок-схема алгоритму Фано

Взагалі існують два апаратних способи щодо керування переповненням буферу [63]. Найбільш надійним з них, з практичної точки зору, є періодичне подавання до кадру заздалегідь відомої послідовності символів, якими зазвичай є нулі, а довжина цього блоку символів співпадає із довжиною кодового обмеження. Якщо за такої умови буфер переповнюється, тоді декодування на цьому етапі вважається незавершеним та невиконаним. У такому разі декодер обнулює буфер, а наступний символ вважається першим символом нового кадру. Тоді процес декодування починається з самого початку. Відповідно, всі дані, які надійшли до декодеру між моментом переповнення буферу та моментом його обнуління, у цьому разі втрачаються. Апаратна реалізація такого підходу дещо знижує швидкість роботи алгоритму декодування та вимагає від інженерів-проектувальників вирішення досить складного завдання синхронізації роботи кодувальної електронної апаратури за часом. Головні особливості часової синхронізації електронних пристроїв розглядалися в першій частині посібника [1].

Головним недоліком описаного способу очищення буферу є те, що у випадку, коли довжина кодового обмеження має не дуже велике значення, його переповнення може статися навіть у випадку правильного проходження послідовності тестових символів. За такої умови частіше використовується інший спосіб, оснований на русі вказівника кадрів вперед. Тоді, у разі знаходження правильного вузла, декодер автоматично переналаштовується, відкидаючи зайві вузли, пройдені у зворотному напрямку. Аналогічний спосіб роботи алгоритму послідовного декодування був описаний на тестовому прикладі у підрозділі 3.8.6.1.

Структурна схема цифрового декодеру згорткового коду, в основі роботи якого лежить алгоритм Фано, показана на рис. 3.87 [63]. Головною особливістю роботи такого декодеру є створення копії кожного кадру кодового слова з використанням кількох допоміжних запам’ятовуючих регістрів. Декодер намагається надсилати символи кодового слова до цих регістрів, створюючи копії коду таким чином, щоб на його виході була кодова

послідовність, близька до переданої, незалежно від можливості спотворення закодованого повідомлення у каналі зв'язку. Відповідність між переданою та прийнятою кодовою комбінацією обчислюється за метрикою різниці між прийнятою та очікуваною кодовими послідовностями. Відповідні параметри декодеру згорткового коду обчислюються з використанням наведених співвідношень (3.331) – (3.337).

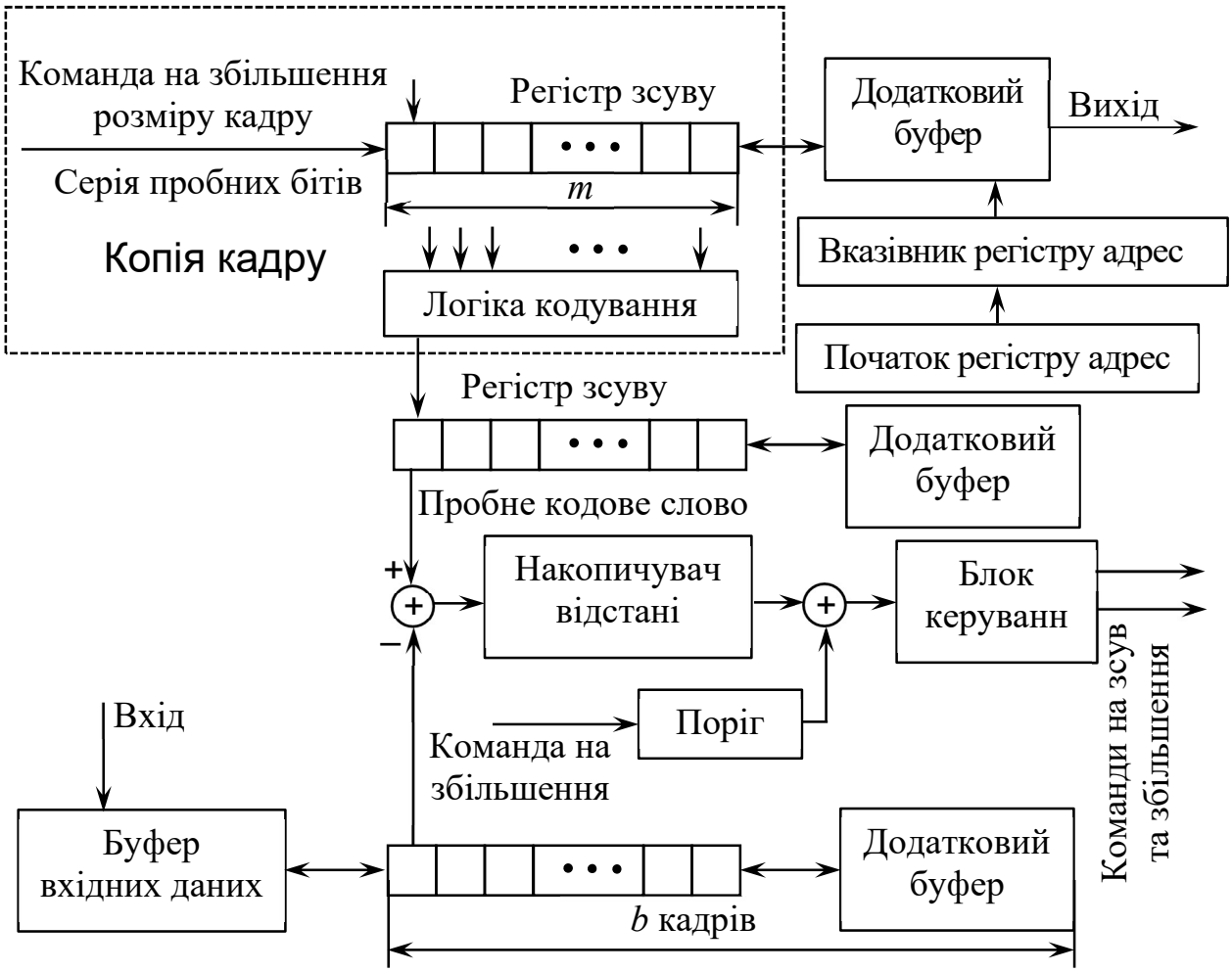


Рис. 3.87 Узагальнена структурна схема цифрового електронного пристрою, призначеного для декодування послідовностей згорткового коду з використанням алгоритму Фано

Після кожної ітерації декодер звертається до останнього прийнятого кадру, який зберігається в одному з запам'ятовуючих регістрів [63]. Тут можливими є наступні операції, пов'язані з поточним опрацюванням кодової

послідовності.

1. Попередній кадр залишається незмінним.

2. Інформації в попередньому кадрі змінюється відповідним чином з метою зниження метрики різниці між прийнятою та очікуваною кодовими послідовностями.

3. Здійснюється повернення до прийнятого раніше кадру.

4. Вводиться новий кадр.

Декодер згорткового коду, структурна схема якого наведена на рис. 3.87, опрацьовує кадри закодованого повідомлення згідно з алгоритмом Фано, блок-схема якого представлена на рис. 3.86, та в процесі цього опрацювання автоматично, на апаратному рівні, обирає одну з чотирьох вищезазначених дій. Проте такий декодер зазвичай є апаратно-програмованим пристроєм, загальний алгоритм його роботи записаний у блоці «Логіка кодування». Зрозуміло, що реалізувати алгоритм Фано безпосередньо на логічних елементах цифрових електронних схем вкрай важко. Зазвичай для запису алгоритму логіки декодування використовують сучасні мікропроцесорні та мікроконтролерні пристрої або програмовані логічні інтегральні схеми (ПЛІС) [1]. У разі використання ПЛІС підвищується швидкодія декодувального пристрою, проте мікропроцесорні та мікроконтролерні засоби зазвичай є більш гнучкими з точки зору можливостей їхнього перепрограмування [1].

Загалом для визначення дій, які необхідно виконати на даній стадії процесу опрацювання кодової послідовності, використовуються наведені співвідношення (3.331) – (3.337), які складають основу алгоритму Фано, блок-схема якого представлена на рис. 3.86. Зрозуміло, що обчислювальні операції головним чином реалізовані в блоці «Логіка кодування», розташованому в системі формування копії кадру. Ці дії формуються автоматично через обчислення критичної відстані згорткового коду $t(l)$ за умови відомого поточного значення порогової величини T. Це значення зберігається в

окремому регістрі «Поріг» та може бути збільшене після надходження відповідної команди. Для пошуку кодової відстані між прийнятою та очікуваними комбінаціями та для сумування кодових відстаней використані логічні схеми «Виключного АБО». Якщо в процесі опрацювання кодової послідовності необхідна її обробка у зворотному напрямку, використовується копія кадра, яка зберігається у відповідному регістрі та опрацьовується через блок «Логіка кодування» відповідної системи [63]. Для уникнення переповнення буферу та для формування посилань на відповідні кадри використовується «Вказівник регістру адрес». Якщо кадр, який оброблявся у зворотному напрямку, вже опрацьований та посилання на нього є зайвим, його можна вивантажити з буфера і це не призводить до помилок алгоритму Фано.

Розглянутий у цьому підрозділі алгоритм Фано загалом має ті ж самі особливості, переваги та недоліки, як і алгоритм послідовного декодування у загальній формі. Вони були досконало описані у підрозділі 3.8.6.4. Тому цілком зрозуміло, що головною перевагою алгоритму Фано є простота обчислювальних процедур та можливість їхньої апаратної реалізації з використанням такої компонентної бази сучасної цифрової електроніки, як ПЛІС, схеми «Виключного АБО» та регістри зсуву. Такі декодери також мають досить високу швидкодію. Головним їх недоліком є можливість переповнення буферу декодеру у разі наявності великої кількості помилок у прийнятій кодовій послідовності [63]. Проте, у разі використання в електронній апаратурі та в системах зв’язку надлишкових згорткових кодів із складною структурою та високою коректувальною здатністю, розглянутий алгоритм Фано зазвичай є більш ефективним, ніж алгоритм Вітербі, оснований на принципі максимальної правдоподібності та розглянутий у підрозділі 3.8.6.3. Головним недоліком використання алгоритму Вітербі для дешифрування таких кодових конструкцій є його вкрай висока обчислювальна складність. Це підтверджують також графічні залежності, наведені у підрозділі 3.8.6.4 на рис. 3.79.

3.8.11 Апаратна реалізація кодерів та декодерів згорткових кодів, призначених для вирішення практичних завдань електроніки

3.8.11.1 Прості цифрові електронні схеми кодерів та декодерів згорткового коду на регістрах зсуву

У підрозділі 3.8.10 була розглянута узагальнена схема цифрового декодеру згорткового коду, в основу роботи якої покладений алгоритм Фано. Ця схема розглядалася з суто теоретичної точки зору. Також зверталась увага на те, що алгоритм Фано, хоча він і є досить простим з обчислювальної точки зору, у сучасній цифровій електронній апаратурі зазвичай реалізується з використанням програмованих пристроїв, тобто мікроконтролерів або ПЛІС.

Проте, згідно із відомими принципами теорії скінченних автоматів, реалізація моделі скінченного автомату можлива як через створення відповідного алгоритму його роботи, так і через електричне з'єднання елементів цифрових електронних схем із пам'яттю, включаючи регістри та тригери [50]. Тому розглянемо у цьому підрозділі приклади електронних схем кодерів та декодерів згорткових кодів, безпосередньо призначених для практичного застосування та побудованих на основі стандартних елементів цифрової електроніки, без використання програмованих пристроїв та засобів програмування електронної апаратури [5]. Головною перевагою таких цифрових електронних схем є їхня вкрай висока швидкодія [8].

Почнемо з простих прикладів апаратної реалізації декодеру згорткового коду з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$, або $n = 2$, $k = 1$ [5].

Приклад 3.64 З використанням регістрів зсуву та схеми «Виключного АБО» побудувати цифрову електронну схему кодеру згорткового коду з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$. Узагальнена схема відповідного кодувального пристрою наведена на рис. 3.61.

Як видно з рис. 3.61, електрична схема такого кодеру може бути безпосередньо побудована на регістрах зсуву та логічних елементах

«Виключного АБО». Відповідна принципова електрична схема із показаними з'єднаннями логічних елементів та регістрів показана на рис. 3.88 [5].

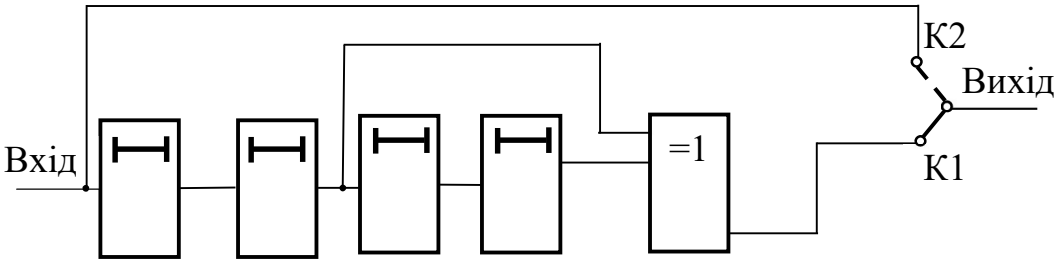


Рис. 3.88 Принципова електрична схема кодеру згорткового коду з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$, побудованого на логічному елементі «Виключного АБО» та на чотирьох регістрах зсуву

Приклад 3.65 З використанням регістрів зсуву та логічних схем «І», «АБО» та «Виключного АБО» побудувати цифрову електронну схему декодеру згорткового коду з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$. Відповідна структурна схема такого кодеру була розглянута у прикладі 3.64 та наведена на рис. 3.88.

Алгоритм роботи цього декодера був розглянутий у підрозділі 3.8.4.2, відповідна схема скінченного автомату наведена на рис. 3.64. Цифрова електронна схема такого декодеру згорткового коду складається з наступних трьох частин.

1. Схема кодувального пристрою.

2. Схема порівняння вхідної кодової послідовності із очікуваною.

3. Схема виправлення помилок.

Аналіз синдрому помилок у згортковому коді здійснюється з використанням поліному синдрому помилок [5], відповідний математичний апарат був розглянутий у підрозділах 3.8.4.1 та 3.8.7.

Цифрова схема декодеру побудована наступним чином [5]. Сигнал з кодувального пристрою сумується за модулем 2 із вхідним сигналом. Слід відзначити, що кодувальний пристрій цілком відповідає схемі кодеру, наведеній на рис. 3.88. Сума сформованого та вихідного сигналів інвертується

та надходить на чотири регістри зсуву. Перший синдром помилки знімається з першого та другого регістрів, а другий – з третього та четвертого. Оскільки символи, які відповідають помилкам, повторюються у векторі синдрому, вони сумуються на логічному елементі «І». Тоді корекція помилки здійснюється автоматично через сумування сигналу, що надходить з логічної схеми «І» із вихідним сигналом, затриманим на два та на чотири такти.

Принципова електрична схема декодеру, який працює за описаним алгоритмом, наведена на рис. 3.89. Вона цілком відповідає узагальненій схемі декодеру, який працює за алгоритмом Фано та був описаний у підрозділі 3.8.10. Ця схема у загальній формі наведена на рис. 3.87.

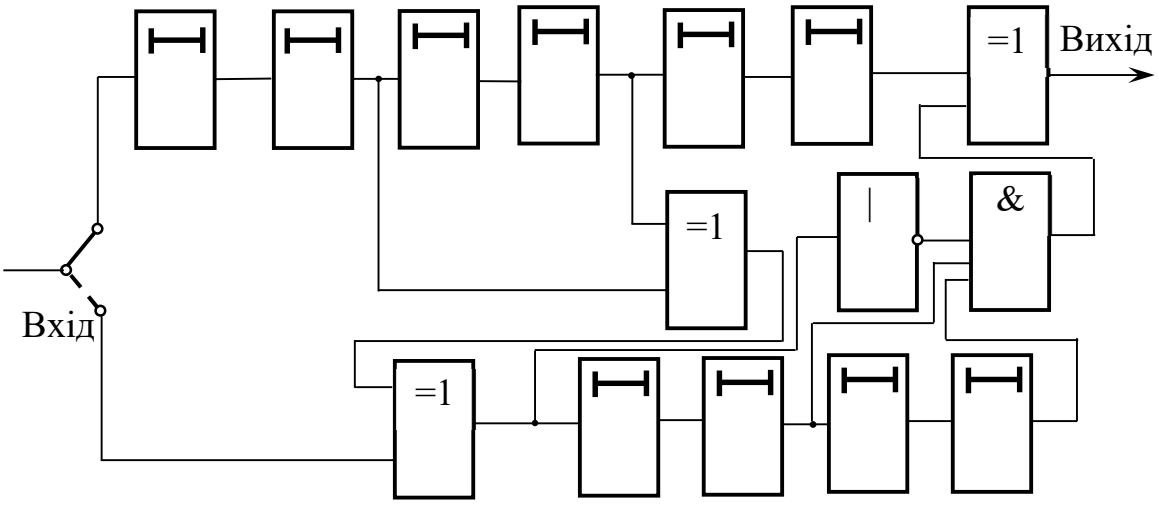


Рис. 3.89 Принципова електрична схема кодеру згорткового коду з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$, побудованого на логічних елементах та регістрах зсуву

3.8.11.2 Апаратна реалізація кодерів та декодерів згорткових кодів, призначених для запису цифрової інформації на магнітні та оптичні носії

У підрозділі 3.7 був розглянутий ітеративний код, призначений для надійного зберігання цифрової інформації на магнітних та оптичних носіях. Було відмічено, що головним принципом формування завадостійкого коду є можливість виправлення пакетних помилок, які виникають на одній доріжці

запису, оскільки імовірність виникнення помилок на різних доріжках в великих кадрах за статистикою є вкрай малою. Зазвичай головною причиною таких помилок є часткові пошкодження магнітного або оптичного покриття [5].

За таких умов згортковий код, призначений для виявлення та виправлення пакетних помилок на магнітних та оптичних носіях, можна побудувати на основі коду Івадаре, узагальнений алгоритм побудови якого був описаний у підрозділі 3.8.9.

Структура згорткового коду, призначеного для виявлення та виправлення пакетних помилок на магнітних на оптичних носіях цифрової інформації, представлена на рис. 3.90 [5]. Головною відмінною рисою цього коду є те, що здійснюється подвійний контроль записаної інформації: за рядками та за діагоналями.

Рис. 3.90 Узагальнена структура згорткового коду, призначеного для контролю помилок запису інформації на магнітних та оптичних носіях

Згідно із способом формування коду Івадаре, описаним у підрозділі 3.8.9, та співвідношеннями (3.327, 3.328), синдромні поліноми $c_{(k)}^{(i)}$ для формування контрольних символів можна записати у наступному вигляді [5]:

$$c^{(1)}_{(k_0+1)}(d) = 1,\ c^{(2)}_{(k_0+1)}(d) = d,\ c^{(3)}_{(k_0+1)}(d) = d^2,\ \ldots,$$
$$c^{(i)}_{(k_0+1)}(d) = d^{i-1},\ \ldots,\ c^{(k_0)}_{(k_0+1)}(d) = d^{k_0-1}, \qquad (3.338)$$

де d – параметр затримки сигналу, який зазвичай використовується як змінна, відносно якої здійснюється поліноміального описання згорткового коду k_0 – загальна кількість доріжок, [5, 63]. За умови виконання співвідношень (3.337) твірні поліноми для створення згорткового коду $g^{(i)}_{(k)}$ записуються наступним чином [5]:

$$g^{(1)}_{(k_0+2)}(d) = d^{n_0-2},\ g^{(2)}_{(k_0+2)}(d) = d^{n_0-2},\ \ldots,$$
$$g^{(i)}_{(k_0+2)}(d) = d^{n_0-2},\ \ldots,\ g^{(k_0)}_{(k_0+2)}(d) = d^{n_0-2}. \qquad (3.339)$$

Структура згорткового коду, побудованого на основі поліномів (3.338), (3.339), є простою і зрозумілою та наочна показана на рис. 3.90. Важливим також є те, що описання структури цього коду не потребує виконання додаткових математичних перетворень.

Щодо визначення величини захисного проміжку, найгіршим є випадок, коли попередній пакет помилок мав місце на першій доріжці, а другий виникає на доріжці з номером k_0, тобто, на останній. Максимальна відстань L між доріжками, помилки на яких будуть виправлені згортковим кодом, визначається співвідношенням [5]:

$$L = (m + 1)n_0 - 1,\ n_0 = k_0 + 2, \qquad (3.340)$$

де m – найбільший степінь твірних поліномів (3.339), n_0 – загальна кількість доріжок. Дійсно, згідно з структурою коду, наочно показаною на рис. 3.90, для запису контрольних символів використовуються саме дві доріжки.

Для апаратної реалізації цифрового електронного пристрою, призначеного для формування такого згорткового коду, необхідно побудувати скінченний автомат із пам’яттю, ємність якої складає n_0k_0 бітів. Відповідна принципова електрична схема кодувального пристрою наведена на рис. 3.91 [5]. Схема складається з D-тригерів, сигнали з яких сумуються на логічних елементах «Виключного АБО». Робота схеми кодування є досить простою та не потребує окремих пояснень.

Проаналізуємо логіку роботи схеми декореру, яка наведена на рис. 3.91 [5].

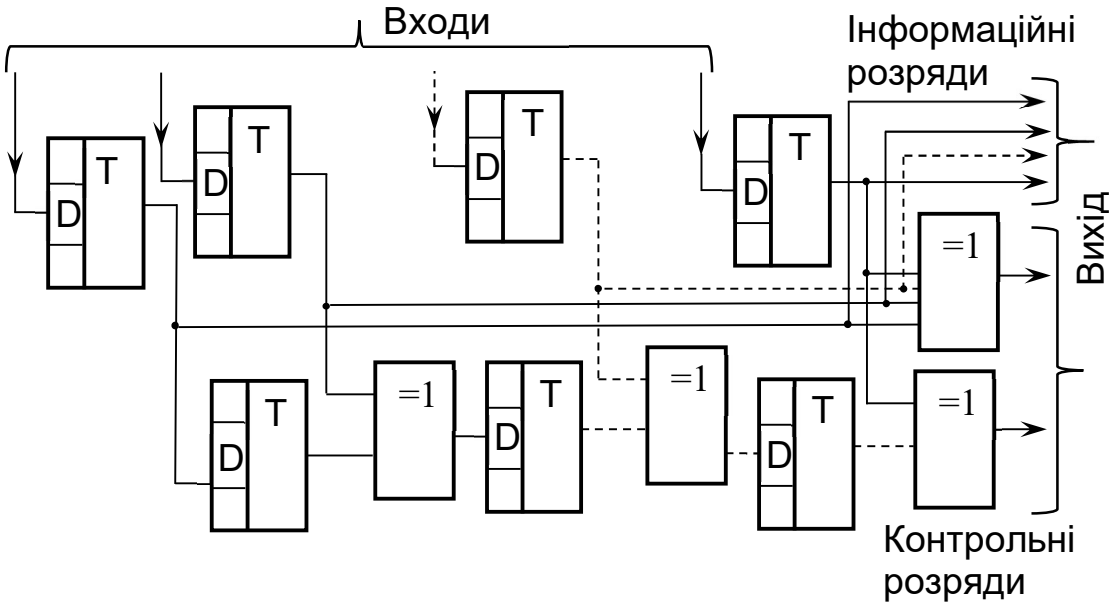


Рис. 3.91 Принципова електрична схема кодеру згорткового коду, призначеного для надійного збереження цифрової інформації на магнітних та оптичних носіях [5]

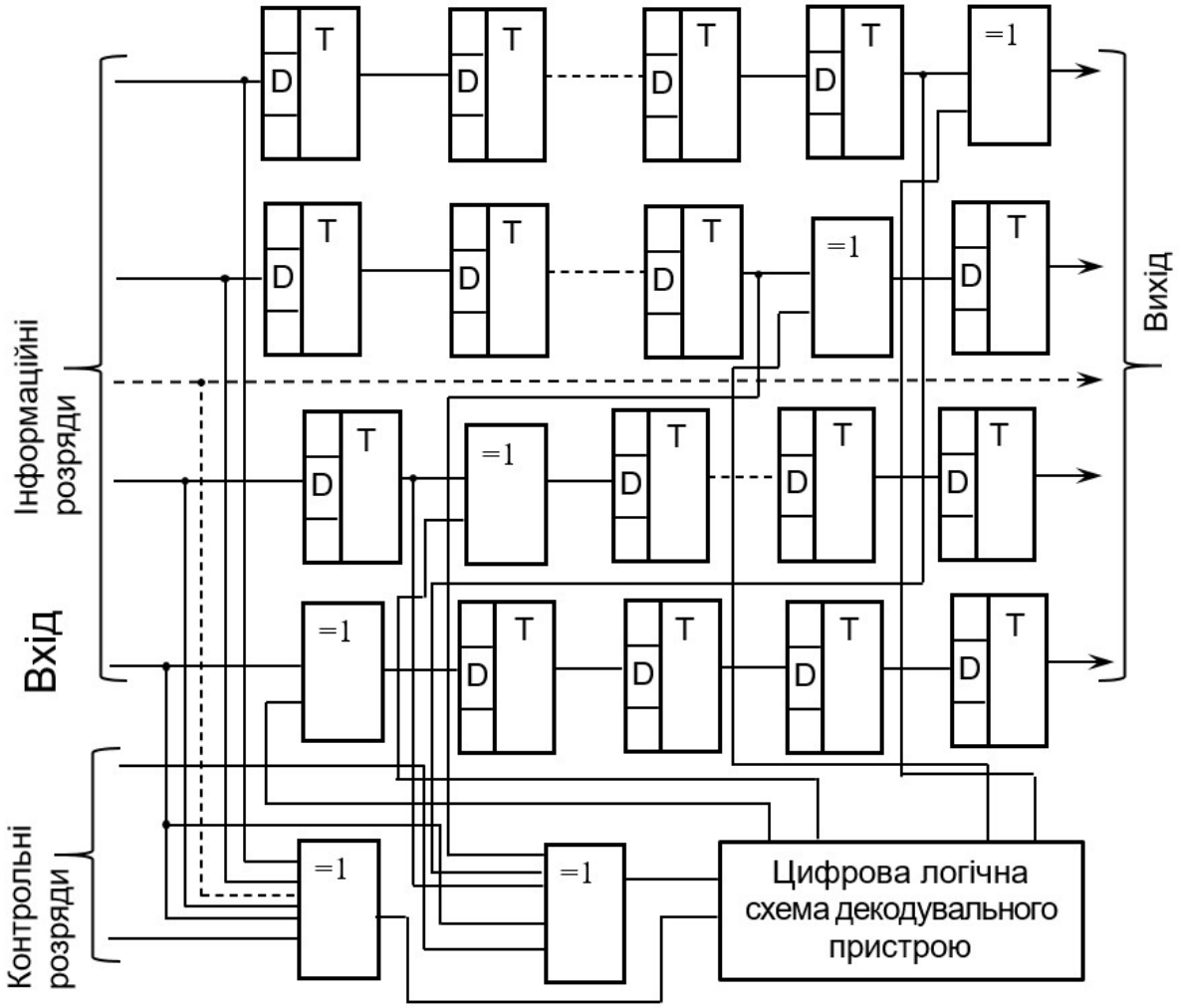


Рис. 3.92 Принципова електрична схема декодеру згорткового коду, призначеного для надійного збереження цифрової інформації на магнітних та оптичних носіях [5]

Закодовані цифрові сигнали, що надходять з магнітного або оптичного записувального електронного пристрою та відповідають послідовностям згорткового коду, послідовно, за тактами, зберігаються в буферних регістрах запам'ятовувального пристрою декодеру. Ці регістри створені на основі D-тригерів та елементів «Виключного АБО» та розташовані в кожному інформаційному каналі за діагональним принципом. Одночасно здійснюється створення символів синдрому помилки $s^{(k_0+1)}(d)$ та $s^{(k_0+2)}(d)$. Символи синдрому $s^{(k_0+2)}(d)$ створюються через додавання за модулем два прийнятих символів рядка, а символи синдрому $s^{(k_0+1)}(d)$ – через додавання символів за діагоналлю. Зрозуміло, що контрольні суми створюються як з інформаційних, так і перевірочних символів. На виході схеми декодеру сформовані сигнали аналізу контрольних сум $s^{(k_0+1)}(d)$ та $s^{(k_0+2)}(d)$ сумуються за модулем два, і, як результат цього сумування, формуються сигнали корекції відповідних бітів [5].

Крім цього, як показано на рис. 3.92, сигнали синдромів помилок за рядками та за діагоналями $s^{(k_0+2)}(d)$ та $s^{(k_0+1)}(d)$ надходять на блок логічної схеми декодувального пристрою, в результаті роботи якого обчислюється номер доріжки та номер кадру, де виникла помилка. Принципова електрична схема логічного блоку декодувального пристрою наведена на рис. 3.93, а алгоритм роботи цього блоку відповідає співвідношенням (3.338).

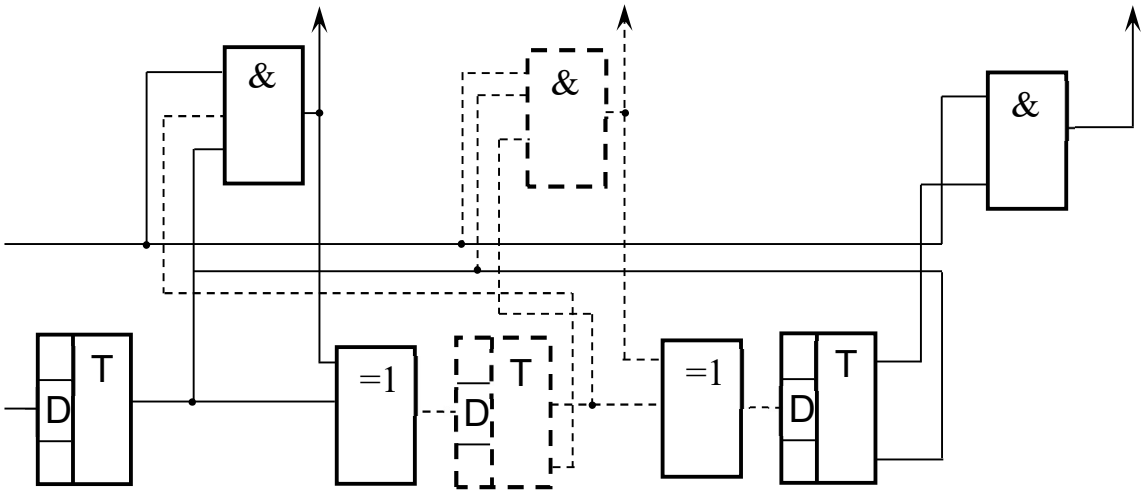


Рис. 3.93 Цифрова логічна схема декодувального пристрою для декодеру згорткового коду, схема якого наведена на рис. 3.92 [5]

Сигнали $s^{(k_0+2)}(d)$, які відповідають синдромам помилки за рядками, надходять на вхід регістру зсуву логічної схеми, і, за умови відсутності сигналу помилки за діагоналлю, вони послідовно проходять вздовж регістру через суматори за модулем два, синхронно з іншими прийнятими сигналами, які відповідають інформаційним символам.

У разі виявлення помилки за діагоналлю формується сигнал, який відповідає ненульовому синдрому контрольної суми $s^{(k_0+1)}(d)$. Цей сигнал надходить на вхід логічної схеми «І», розташованої в схемі декодування. Оскільки, за логікою формування коду, вважається, що рядок містить лише одну помилку, сигнал помилки, створений після формування сигналу корекції, вже не надходить на інші регістрі зсуву, які розташовані далі.

Сформовані сигнали корекції надходять на відповідні входи логічних схем «Виключного АБО», які розташовані в кожному каналі декодеру, і в результаті пакетні помилки на доріжках автоматично виправляються в режимі реального часу [5]. Описана у цьому підрозділі кодувальна система ефективно використовуєтсья в сучасних електронних системах, призначених для запису та читання інформації з магнітних та оптичних носіїв [5].

3.8.12 Програмна реалізація кодерів та декодерів згорткових кодів з різними параметрами

3.8.12.1 Матриці станів згорткового коду та їх алгоритмічне описання через числові тривимірні масиви

Як було відмічено у підрозділі 3.8.10, більшість методів аналізу згорткових кодів орієнтовані не на апаратну, а на програмну реалізацію, оскільки як алгоритм послідовного декодування, так і алгоритм Вітербі, є досить складними для апаратної реалізації у вигляді цифрових електронних схем. Хоча алгоритм послідовного декодування є ітераційним та досить простим, але його ефективна апаратна реалізація вкрай ускладняється можливістю переповнення буферу даних у разі необхідності виконання великої кількості ітерацій у зворотному напрямку [63]. Щодо алгоритму Вітербі, який у загальній формі був описаний у підрозділі 3.8.6.3 та на перший

погляд є досить є простим та зрозумілим, його формалізація з обчислювальної точки зору можлива лише з використанням сучасних методів теорії оптимізації, зокрема функцій лінійного програмування [63]. Тобто, розробка декодерів згорткових кодів як суто апаратних електронних пристроїв, побудованих на схемах цифрової логіки та описаних у підрозділі 3.8.11, є можливою лише для простих кодових конструкцій. Крім цього, несумлінною перевагою програмних реалізацій алгоритмів кодування та декодування згорткових кодів є їх висока гнучкість та можливість легкої зміни алгоритму роботи кодувальної апаратури з метою його адоптації для виконання реальних практичних завдань.

Узагальненим математичним методом описання структури згорткових кодів є теорія скінченних автоматів, описана у третьому томі другої частини посібника [50]. Головні особливості описання алгоритмів формування згорткових кодів з точки зору теорії скінченних автоматів розглядалися в підрозділі 3.8.4.2. Щодо математичних моделей, пов'язаних із реалізацією алгоритму Вітербі [63], вони є досить складними та виходять за рамки цього посібника.

Алгоритми формування згорткового коду є досить простими, вони безпосередньо базуються на використанні теорії скінченних автоматів та логічних функцій дискретної математики та алгебри Буля [48, 50], зокрема функції сумування двійкових чисел за модулем два. Реалізація таких алгоритмів обробки та формування двійкових числових послідовностей в комп'ютерних програмах є досить простою та не має суттєвих лінгвістичних особливостей [50]. Для деяких структур згорткового коду вона буде розглянута в підрозділі 3.8.12.3.

Навпаки, складні декодувальні алгоритми, основою для формування яких також є теорія скінченних автоматів, потребують окремого розгляду з точки зору формалізації їхнього описання та пристосування до теорії регулярних мов комп'ютерної арифметики та до сучасних засобів програмування [50]. Наприклад, описання алгоритму Фано, наведене у підрозділі 3.8.10, є досить узагальненим та не повністю формалізованим, хоча блок-схема алгоритму, наведена на рис. 3.86, дозволяє провести таку формалізацію для розв'язування відповідних практичних завдань.

Проте сама теорія скінченних автоматів, як універсальний сучасний інструмент для аналізу обчислювальних алгоритмів дискретної математики [50], в даному випадку також вказує досить простий шлях щодо пошуку загального методу формалізації алгоритму розшифрування конструкцій згорткових кодів.

Сформулюємо спочатку завдання декодування конструкцій згорткового коду у загальному вигляді як пошук шляху із мінімальною метрикою від кореня ґратки деревоподібної діаграми станів до її вершини. У такій загальній формі метод визначення найбільш правдоподібної комбінації згорткового коду через мінімальну метрику шляху за Хеммінгом був описаний у підрозділі 3.8.5. Тоді для формалізації, з використанням числових алгоритмів, описання структури ґратки, з метою подальшого формування формалізованого методу щодо пошуку на ній шляху з найменшою метрикою, може бути ефективно використаний математичний апарат матричного аналізу, який був описаний у третьому томі другої частини посібника [50].

Щоб розв'язати поставлене завдання, опишемо спочатку діаграму станів скінченного автомату в матричній формі. Матрицю, яку надалі будемо називати матрицею станів згорткового коду, сформуємо за наступними правилами, які сформулюємо як загальні властивості цієї матриці.

Властивість 3.18. Матриця станів **M** завжди є тривимірною.

Властивість 3.19. Кожний стовпець матриці **M** визначає поточний стан скінченного автомату, який формує згортковий код, а відповідні рядки – наступний стан, перехід до якого є можливим за відповідних умов.

Властивість 3.20. На першій сторінці матриці станів у колонці i, яка визначає номер поточного стану, та у рядку j, де j – номер можливого наступного стану скінченного автомату, стоїть значення вхідного цифрового сигналу m, у разі дії якого здійснює перехід від стану i до стану j. Для двійкових цифрових сигналів значення m завжди дорівнює 0 або 1.

Властивість 3.21. Якщо перехід із стану i до стану j є неможливим, відповідний елемент першої сторінки матриці станів $\mathbf{M}(j, i, 1)$ дорівнює –1.

Властивість 3.22. Якщо елемент матриці станів $\mathbf{M}(j, i, 1)$, розташований на першій сторінці, дорівнює –1, тоді елементи цієї матриці з тим же самим рядком та стовпчиком j та i на інших сторінках також дорівнюють –1. З використанням мови предикатів сформульоване правило можна переписати наступним чином [50]:

$$\forall i, j, l\big((\mathbf{M}(j, i, 1) = -1) \rightarrow (\mathbf{M}(j, i, l) = -1)\big). \quad (3.341)$$

Властивість 3.23 (зворотна до властивості 3.22). Якщо елемент матриці станів **M**(*j*, *i*, 1), розташований на першій сторінці, не дорівнює –1, тоді елементи цієї матриці з тим же самим рядком та стовпчиком *j* та *i* на інших сторінках також не дорівнюють –1. З використанням мови предикатів сформульоване правило можна переписати наступним чином [50]:

$$\forall i, j, l\big((\mathbf{M}(j, i, 1) \neq -1) \rightarrow (\mathbf{M}(j, i, l) \neq -1)\big). \quad (3.342)$$

Наступні властивості визначають розмірність матриці **M** та її загальну структуру.

Властивість 3.24 Для згорткового коду з довжиною кодового обмеження *K* та коефіцієнтом надлишковості $\frac{1}{n}$ розмірність матриці станів **M** складає (2^{K-1}, 2^{K-1}, n + 1), де $s = 2^{K-1}$ – загальна кількість станів скінченного автомату.

Властивість 3.25 Згідно із властивостями 3.20 та 3.21, на першій сторінці матриці **M** записуються значення вхідного цифрового сигналу, які відповідають здійсненню переходу із стану *i* до стану *j*, де *i* – номер стовпчика, а *j* – номер рядка. Наступні сторінки матриці **M** описують, за тієї ж умови, значення вихідного цифрового сигналу кодувального пристрою. Першому розряду вихідного сигналу відповідає елемент другого стовпчика матриці станів **M**(*j*, *i*, 2), другому розряду – елемент **M**(*j*, *i*, 3), розряду з довільним номером *l* – елемент **M**(*j*, *i*, *l* + 1), а останньому розряду – відповідно, елемент **M**(*j*, *i*, *n* + 1). Згідно із властивістю 3.22 та співвідношенням (3.341), якщо перехід від стану *i* до стану *j* є неможливим, значення елемента матриці **M**(*j*, *i*, *l*) дорівнює –1 за будь-яких значень *l*.

Слід відзначити, що під час формування матриці станів, як і взагалі в теорії кодування сигналів, важливим є порядок нумерації розрядів у вихідному сигналі кодеру, який може бути прямим або зворотним. Зазвичай використовують прямий порядок, тому у разі, якщо має місце зворотній, розряди слід автоматично перенумерувати.

Узагальнена структура матриці станів, яка описується властивостями 3.18 – 3.25, наочно показана на рис. 3.94. Розглянемо її більш досконало. Для стану кодеру з номером *i* перший розряд вихідної кодової комбінації позначимо як $\mathbf{U}_{i,1}$, другий – $\mathbf{U}_{i,2}$, а останній – $\mathbf{U}_{i,n}$. Параметр 0 на рис. 3.94 означає нульове значення вхідного сигналу, індекс *l* – номер розряду вихідної

кодової комбінації, а індекс i – номер стану кодеру. Наприклад, запис $\mathbf{U}_{i,2}(0)$ означає, що у відповідному елементі матриці станів записаний другий розряд вихідного сигналу за умови початкового стану кодеру з номером i та нульового значення вхідного сигналу. Згідно із властивістю 3.20 визначаємо, що, за логікою роботи скінченного автомату, у разі дії цього вхідного сигналу кодер переходить із стану i до стану j, де i – номер стовпчика, а j – номер рядка матриці станів. А номер сторінки матриці, де розташоване значення другого розряду вихідного сигналу, визначається за властивістю 3.25 та становить $2 + 1 = 3$. Тобто, значенню вихідного сигналу $\mathbf{U}_{i,2}(0)$ відповідає елемент матриці $\mathbf{M}(j, i, 3)$, де i – початковий стан кодеру, а j – стан, до якого він переходить за умови дії нульового вхідного сигналу. Відповідно, за цих умов $\mathbf{M}(j, i, 1) = 0$.

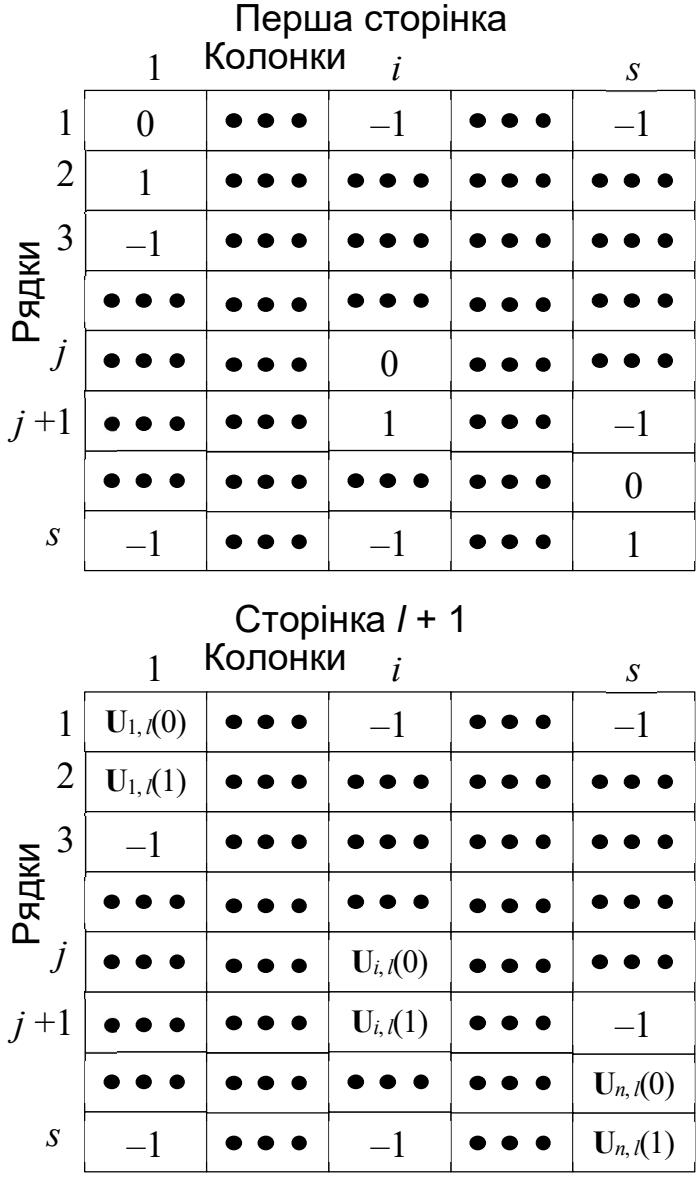


Рис. 3.94 Узагальнена структура матриці станів. Відображена перша сторінка матриці та сторінка з номером $l + 1$

З розглянутої структури матриці станів та описаних її властивостей можна зрозуміти, що на першій сторінці цієї матриці лише два елементи будь-якого стовпчика i не дорівнюють –1, а саме елементи $\mathbf{M}(j_0, i, 1)$ та $\mathbf{M}(j_1, i, 1)$. Тут індекс j_0 відповідає номеру стану, в який переходить кодер зі стану i у разі дії нульового вхідного сигналу, а індекс j_1 – номеру стану, в який він переходить зі стану i у разі дії одиничного вхідного сигналу. Тому зрозуміло, що $\mathbf{M}(j_0, i, 1) = 0$, $\mathbf{M}(j_1, i, 1) = 1$. Також із властивості 3.21 можна зробити висновок, що всі інші елементи у стовпчику i дорівнюють –1, або, на мові теорії предикатів:

$$\forall i, j\big((i \in 1 \dots 2^n, j \neq j_0, j \neq j_1) \rightarrow (\mathbf{M}(j, i, 1) = -1)\big). \qquad (3.343)$$

Тоді, згідно із властивістю 3.23, елементи стовпчика i на всіх сторінках матриці станів також дорівнюють –1, виключенням є лише ті елементи, які розташовані у рядках з номерами j_0 та j_1.Зрозуміло, що у рядку j_0 посторінково розташовані біти вихідного сигналу кодеру $\mathbf{U}$, який є результатом дії вхідного сигналу $m = 0$. Аналогічно, у рядку j_1 посторінково розташовані біти вихідного сигналу кодеру, який формується у разі $m = 1$. Тобто, на мові теорії предикатів:

$$\forall i, j, l\left(\exists(i \in 1 \dots 2^n, j = j_0) \rightarrow \left(\mathbf{M}(j_0, i, l) = \mathbf{U}_{i,l-1}(0)\right)\right),$$

$$\forall i, j, l\left(\exists(i \in 1 \dots 2^n, j = j_1) \rightarrow \left(\mathbf{M}(j_1, i, l) = \mathbf{U}_{i,l-1}(1)\right)\right). \qquad (3.344)$$

Використання матриці станів для розшифрування конструкцій згорткових кодів дозволяє формалізувати алгоритм пошуку оптимального шляху за деревоподібною ґраткою діаграми станів. Відповідний алгоритм може бути реалізований з використанням відомих сучасних засобів матричного програмування системи науково-технічних розрахунків MatLab [13, 14].

3.8.12.2 Приклади формування матриці станів згорткового коду

Розглянемо приклади формування матриці станів для деяких конструкцій згорткових кодів з різними параметрами.

Приклад 3.66 Побудувати матрицю станів згорткового коду з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$. Провести аналіз елементів побудованої матриці. Відповідна структурна схема кодеру наведена на рис. 3.61.

Аналіз роботи цього кодувального пристрою був проведений у

підрозділі 3.8.3. Кодер має буферний регістр на три біти, тому скінченний автомат, який йому відповідає, описується чотирма станами: 00, 01, 10 та 11. Результати реакції цього кодувального пристрою на вхідні сигнали 0 та 1 для різних станів системи представлені на рис. 3.62, а відповідна схема скінченного автомату наведена на рис. 3.64.

Для такої кодувальної схеми матриця станів має розмірність (4, 4, 3). Структура цієї матриці посторінково, у вигляді трьох двовимірних масивів із розмірністю (4, 4), представлена у таблиці 3.30.

Таблиця 3.30 – Посторінкове подання матриці станів для згорткового коду з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$

Перша сторінка	Друга сторінка	Третя сторінка
$\begin{pmatrix} 0 & -1 & 0 & -1 \\ 1 & -1 & 1 & -1 \\ -1 & 0 & -1 & 0 \\ -1 & 1 & -1 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & -1 & 1 & -1 \\ 1 & -1 & 0 & -1 \\ -1 & 1 & -1 & 0 \\ -1 & 0 & -1 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & -1 & 1 & -1 \\ 1 & -1 & 0 & -1 \\ -1 & 0 & -1 & 1 \\ -1 & 1 & -1 & 0 \end{pmatrix}$

Проаналізуємо деякі елементи тривимірної матриці, наведеної у таблиці 3.30. На першій сторінці в першому стовпчику, який позначимо як матрицю $\mathbf{M_1}$, не дорівнюють –1 лише елементи першого та другого рядків. Це означає, що зі стану 1 система може перейти до стану 2, або залишатися в ньому певний період часу. Елемент $\mathbf{M_1}(1,1) = 0$, що свідчить про те, що система залишається в першому стані за умови надходження вхідного сигналу $m = 0$. Елемент матриці $\mathbf{M_1}(2,1) = 1$, і це означає, що система переходить зі стану 1 до стану 2 у разі надходження вхідного сигналу $m = 1$. Для аналізу відповідних вихідних сигналів необхідно перейти до другої та третьої сторінок матриці **M**, які позначимо як $\mathbf{M_2}$ та $\mathbf{M_3}$. Оскільки $\mathbf{M_1}(1,1) = 0$ та $\mathbf{M_2}(1,1) = 0$, можна зробити висновок про те, що якщо кодер знаходиться у стані 1 і надходить вхідний сигнал $m = 0$, формується послідовність бітів вихідного сигналу 00. Значення елементів матриці станів $\mathbf{M_1}(2,1)$ та $\mathbf{M_2}(2,1)$ також є однаковими та дорівнюють 1. Це означає, що за умови надходження на вхід сигналу 1 кодер, переходячи з

першого стану до другого, формує вихідну кодову послідовність 11.

Аналіз другого стовпчику першої сторінки матриці станів показує, що елемент $\mathbf{M_1}(3,2) = 0$. Це означає, що у разі надходження вхідного сигналу $m = 0$ кодер переходить з другого стану до третього. Проаналізуємо, який вихідний сигнал відповідає цьому переходу. Оскільки $\mathbf{M_2}(3,2) = 1$, а $\mathbf{M_3}(3,2) = 0$, буде сформована вихідна кодова послідовність 10. Зверніть особливу увагу на те, що перший біт вихідної кодової послідовності $\mathbf{U}_{2,1}(0) = 1$ стоїть зліва, а другий біт, $\mathbf{U}_{2,2}(0) = 0$ – справа. Тобто, у даному випадку використаний прямий запис числової послідовності. Інший елемент другого стовпчика першої сторінки матриці станів, який не дорівнює –1, це елемент $\mathbf{M_1}(4,2) = 1$. Тобто, за умови дії вхідного сигналу $m = 0$ кодер переходить з другого стану до четвертого. Оскільки $\mathbf{M_2}(4,2) = 0$, а $\mathbf{M_3}(4,2) = 1$, у цьому випадку формується вихідна кодова послідовність 01.

Аналогічно проводиться аналіз третього та четвертого стовпчиків матриці станів. Такий аналіз показує, що матриця станів, наведена у таблиці 3.30, цілком відповідає алгоритму роботи скінченного автомату, структурна схема якого наведена на рис. 3.64.

Приклад 3.67 Побудувати матрицю станів згорткового коду з параметрами $K = 5$, $\frac{1}{n} = \frac{1}{3}$. Провести аналіз елементів побудованої матриці для одного з її стовпчиків. Відповідна структурна схема кодеру наведена на рис. 3.69.

Алгоритм роботи кодеру з довжиною кодового обмеження $K = 5$ та коефіцієнтом надлишковості $\frac{1}{3}$, структурна схема якого наведена на рис. 3.69, був досить ретельно проаналізований у підрозділі 3.8.5. Можливі переходи між станами кодеру показані у таблиці 3.27, а на рис 3.70 наведена діаграма станів кодеру у вигляді схеми скінченного автомату. Тому у даному випадку будемо будувати матрицю станів, яка описує скінченний автомат, структурна схема якого наведена на рис. 3.70. За такої умови можна також використовувати інформацію про можливі переходи між станами скінченного цього автомату, яка наведена у таблиці 3.27.

Для спрощення формування матриці станів замість імен станів кодеру, використаних у таблиці 3.27 та на рис. 3.70, використаємо їхню нумерацію в

десятковому форматі запису чисел. Тобто, введемо відповідні заміни:

$$a \to 1, b \to 2, c \to 3, d \to 4, e \to 5, f \to 6, g \to 7, h \to 8, i \to 9,$$
$$j \to 10, k \to 11, l \to 12, n \to 14, o \to 15, p \to 16. \qquad (3.345)$$

Розмірність матриці станів визначимо з використанням властивості 3.24. Зрозуміло, що для конструкції згорткового коду, яка розглядається, кількість рядків та стовпчиків матриці становить $2^{5-1} = 16$, що відповідає співвідношенню (3.345). Щодо кількості сторінок матриці, вона, згідно із властивістю 3.24, складає $l = n + 1 = 4$. Тобто, загальна розмірність матриці станів, яка формується, складає (16 x 16 x 4), що відповідає кількості елементів 1024.

На першу сторінку матриці станів будемо записувати значення вхідного сигналу m, у разі якого кодер переходить зі стану i до стану j, де i – номер стовпчика матриці, j – номер рядка, а на другій, третій та четвертій сторінках – відповідно перший, другий та третій біти вихідного сигналу, який відповідає цій зміні станів. Відповідні числові дані наведені в таблиці 3.27. У разі, якщо переходу між станами з відповідними номерами не існує, цей елемент матриці станів, згідно із властивістю 3.21, дорівнює –1.

Матриця станів, яка відповідає таблиці 3.27, у посторінковому вигляді представлена на рис. 3.95.

Перша сторінка матриці

$$\begin{pmatrix}
0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 0 \\
-1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1
\end{pmatrix}$$

Рис. 3.95 Матриця станів згорткового коду для прикладу 3.67

Друга сторінка матриці

$$\begin{pmatrix}
0&-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1\\
1&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1&-1&-1\\
-1&1&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1&-1\\
-1&0&-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1\\
-1&-1&1&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1\\
-1&-1&0&-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1\\
-1&-1&-1&0&-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1\\
-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1&-1\\
-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1\\
-1&-1&-1&-1&0&-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1\\
-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1&-1&-1&1&-1&-1\\
-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&0&-1&-1\\
-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&0&-1\\
-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&1&-1\\
-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&0\\
-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&1
\end{pmatrix}$$

Третя сторінка матриці

$$\begin{pmatrix}
0&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1&-1&-1\\
1&-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1\\
-1&1&-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1\\
-1&0&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1&-1\\
-1&-1&1&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1\\
-1&-1&0&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1\\
-1&-1&-1&0&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1&-1\\
-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1\\
-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1\\
-1&-1&-1&-1&0&-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1\\
-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1&-1&-1&1&-1&-1\\
-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&0&-1&-1\\
-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&0&-1\\
-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&1&-1\\
-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&0\\
-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&1
\end{pmatrix}$$

Четверта сторінка матриці

$$\begin{pmatrix}
0&-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1\\
0&-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1\\
-1&1&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1&-1\\
-1&1&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1&-1\\
-1&-1&1&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1\\
-1&-1&1&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1\\
-1&-1&-1&0&-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1\\
-1&-1&-1&0&-1&-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1\\
-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1\\
-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&0&-1&-1&-1\\
-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1&-1&-1&1&-1&-1\\
-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1&-1&-1&1&-1&-1\\
-1&-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1&-1&-1&1&-1\\
-1&-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1&-1&-1&1&-1\\
-1&-1&-1&-1&-1&-1&1&-1&-1&-1&-1&-1&-1&-1&-1&1\\
-1&-1&-1&-1&-1&-1&0&-1&-1&-1&-1&-1&-1&-1&-1&0
\end{pmatrix}$$

Рис. 3.95 Матриця станів згорткового коду для прикладу 3.67 (закінчення)

Проаналізуємо отриману матрицю станів, сторінки якої наведені на рис. 3.95, для одного із станів кодеру, наприклад, для стану 11. Із співвідношень (3.345) видно, що стан з цим номером за позначеннями, використаними у таблиці 3.27, відповідає стану *k*. Згідно з таблицею 3.27 із стану *k* кодер може перейти або до стану *e* у разі дії вхідного сигналу $m = 0$, або до стану *f* за умови дії вхідного сигналу $m = 1$. Також, згідно з інформацією, наведеною у таблиці 3.27, переходу $k \rightarrow e$ відповідає вихідний сигнал кодеру **U** = 000, а переходу $k \rightarrow f$ – вихідний сигнал **U** = 001. Особливу увагу необхідно звернути на те, що біти вихідного сигналу записані у прямому порядку, тобто, перший біт кодової послідовності розташований справа, а останній –зліва. Згідно із співвідношеннями (3.345) стану *e* відповідає номер 5, а стану *f* – номер 6.

Розглянемо тепер стовпчик 11 першої сторінки матриці станів, наведеної на рис.3.95. Видно, що всі елементи цього стовпчику, крім п'ятого та шостого, дорівнюють –1, що згідно з властивістю 3.21, свідчить про те, що переходи до цих станів із стану 11 є неможливими. Що до п'ятого та шостого елементів одинадцятого стовпчика, їх значення є іншими та становлять, відповідно, $\mathbf{M}(5, 11, 1) = 0$ та $\mathbf{M}(6, 11, 1) = 1$. Це означає, що у разі дії вхідного сигналу $m = 0$ кодер переходить із стану 11 до стану 5, а у разі дії вхідного сигналу $m = 1$ – із стану 11 до стану 6. Тобто, числові дані, які записані у стовпчику 11 першої сторінки матриці станів, цілком відповідають інформації, яка наведена у таблиці 3.27.

Розглянемо тепер другу, третю та четверту сторінку матриці станів, наведеної на рис. 3.95. Як і на першій сторінці цієї матриці, будемо аналізувати п'ятий та шостий елементи одинадцятого стовпчика, які відповідають бітам вихідного сигналу, що формується кодером. Із рис. 3.95 видно, що $\mathbf{M}(5, 11, 2) = 0$, $\mathbf{M}(6, 11, 2) = 1$, $\mathbf{M}(5, 11, 3) = 0$, $\mathbf{M}(6, 11, 3) = 0$, $\mathbf{M}(5, 11, 4) = 0$, $\mathbf{M}(6, 11, 4) = 0$. Зверніть увагу на те, що інші елементи одинадцятого стовпчика матриці станів на всіх сторінках дорівнюють –1. Тобто, інформація щодо бітів вихідного сигналу, яка наведена у таблиці 3.27, також правильно відображена у створеній матриці станів.

Аналогічно можна перевірити будь-який інший стовпчик матриці станів та впевнитись, що всі її стовпчики цілком відповідають інформації, яка

наведена у таблиці 3.27.

Приклад 3.68 Побудувати матрицю станів згорткового коду з параметрами $K = 4$, $\frac{1}{n} = \frac{1}{3}$. Структурна схема кодеру наведена на рис. 3.96.

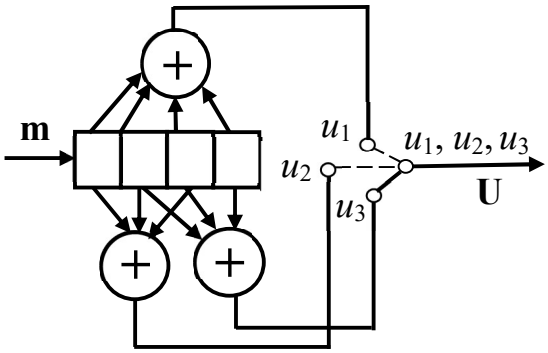


Рис. 3.96 Структурна схема кодеру згорткового коду з параметрами $K = 4$, $\frac{1}{n} = \frac{1}{3}$, для прикладу 3.68

Для вирішення поставленої задачі спочатку створимо таблицю, аналогічну таблиці 3.27, в якій відобразимо всі можливі переходи між станами кодеру. Відповідні числові дані наведені в таблиці 3.31.

Таблиця 3.31 – Можливі переходи між станами схеми формування згорткового коду, наведеної на рис. 3.96

Поточний стан	Вхідний сигнал	Стан, до якого переходить кодер	Вихідний сигнал
1. $a = 000$	$m = 0$	$a \rightarrow a = 000$	**U** = 000
	$m = 1$	$a \rightarrow b = 100$	**U** = 110
2. $b = 100$	$m = 0$	$b \rightarrow c = 010$	**U** = 111
	$m = 1$	$b \rightarrow d = 110$	**U** = 001
3. $c = 010$	$m = 0$	$c \rightarrow e = 001$	**U** = 111
	$m = 1$	$c \rightarrow f = 101$	**U** = 001
4. $d = 110$	$m = 0$	$d \rightarrow g = 011$	**U** = 000
	$m = 1$	$d \rightarrow h = 111$	**U** = 110
5. $e = 001$	$m = 0$	$e \rightarrow a = 000$	**U** = 101
	$m = 1$	$e \rightarrow b = 100$	**U** = 011
6. $f = 101$	$m = 0$	$f \rightarrow c = 010$	**U** = 010
	$m = 1$	$f \rightarrow d = 110$	**U** = 100

Таблиця 3.31 (продовження)

Поточний стан	Вхідний сигнал	Стан, до якого переходить кодер	Вихідний сигнал
7. $g = 011$	$m = 0$	$g \rightarrow e = 001$	**U** = 010
	$m = 1$	$g \rightarrow f = 101$	**U** = 100
8. $h = 111$	$m = 0$	$h \rightarrow g = 011$	**U** = 101
	$m = 1$	$h \rightarrow h = 111$	**U** = 011

На рис. 3.97 представлена схема скінченного автомату, який відповідає даним, наведеним у таблиці 3.31 та, відповідно, описує алгоритм роботи схеми кодеру, наведеної на рис. 3.96.

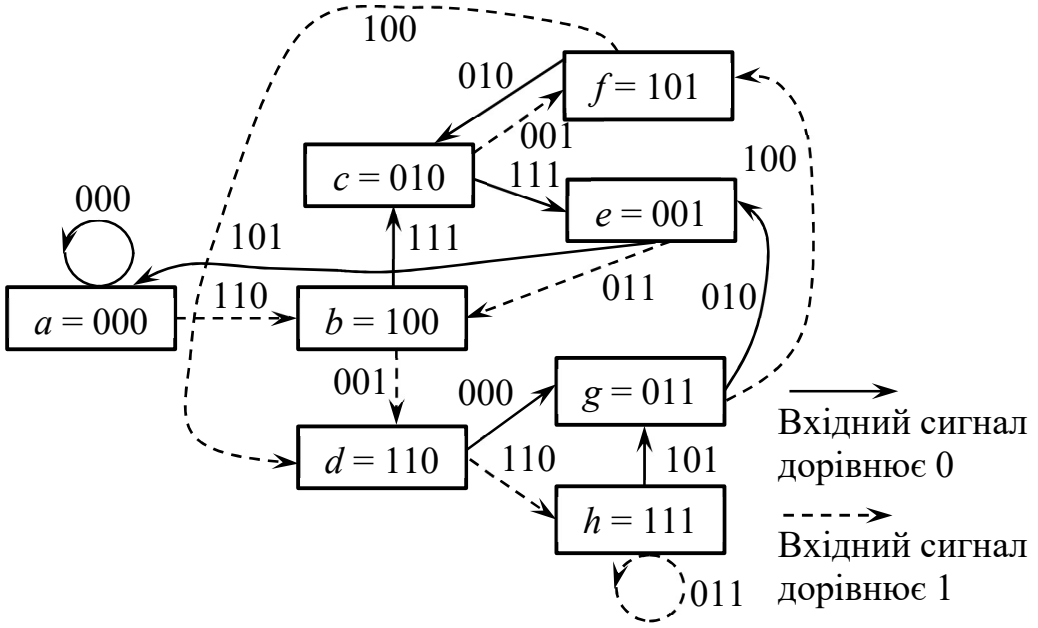


Рис. 3.97 Схема скінченного автомату, який відповідає структурній схемі кодеру згорткового коду з параметрами $K = 4$, $\frac{1}{n} = \frac{1}{3}$, наведеній на рис. 3.96

За числовими даними, наведеними у таблиці 3.31, можна легко побудувати матрицю станів, яка буде точно описувати алгоритм роботи схеми кодеру згорткового коду, наведеної на рис. 3.96. Цей алгоритм цілком відповідає схемі скінченного автомату, яка наведена на рис. 3.97. Спосіб побудови матриці станів за умови відомих переходів між станами кодеру був

описаний у підрозділі 3.8.12.1 та розглянутий для цифрових схем, які створюють згорткові коди з іншими параметрами, у прикладах 3.66 та 3.67.

Для кодеру з параметрами $K = 4$, $\frac{1}{n} = \frac{1}{3}$, який розглядається в цьому прикладі, розмірність матриці станів складає $(2^{4-1}, 2^{4-1}, 3+1) = (8 \times 8 \times 4)$. Структура матриці станів цілком відповідає числовим даним, наведеним у таблиці 3.31. Ця матриця посторінково відображена у таблиці 3.32.

Таблиця 3.32 – Посторінкове подання матриці станів для згорткового коду з параметрами $K = 4$, $\frac{1}{n} = \frac{1}{3}$

Перша сторінка	Друга сторінка
$\begin{pmatrix} 0 & -1 & -1 & -1 & 0 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & 0 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 0 & -1 & -1 & -1 & 0 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & 0 & -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 0 & -1 & -1 & -1 & 1 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 & 0 \end{pmatrix}$
Третя сторінка	**Четверта сторінка**
$\begin{pmatrix} 0 & -1 & -1 & -1 & 0 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & 0 & -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & 0 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & -1 & -1 & 0 \\ -1 & -1 & -1 & 1 & -1 & -1 & -1 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ 0 & -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & -1 & -1 & 1 \\ -1 & -1 & -1 & 0 & -1 & -1 & -1 & 1 \end{pmatrix}$

Матриці стану, отримані у прикладах 3.66 – 3.68, будуть використанні в комп'ютерній програмі, призначеній для декодування послідовностей згорткових кодів із заданими параметрами та для виправлення в них помилкових розрядів. Відповідні алгоритми декодування, основані на використанні аналізу отриманих матриць стану, а також особливості реалізації цих алгоритмів з використанням засобів матричного програмування системи науково-технічних розрахунків MatLab [13, 14], розглядатимуться далі у підрозділах 3.8.12.4 та 3.8.12.5.

3.8.12.3 Узагальнений алгоритм формування послідовностей згорткового коду

Алгоритм формування згорткових кодів реалізувати досить просто. Для цього, насамперед, необхідно знати тип коду та його параметри, а саме, довжину кодового обмеження K та коефіцієнт надлишковості $\frac{1}{n}$. За відомою схемою кодеру, через значення бітів b_i буферного регістру B формуються контрольні суми та, відповідно, біти вихідного сигналу. Наприклад, для кодеру з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$, структурна схема якого наведена на рис. 3.61, біти вихідного цифрового сигналу записуються наступним чином:

$$u_1 = \mathrm{mod}_2(b_1, b_3); \qquad u_2 = \mathrm{mod}_2(b_1, b_2, b_3). \tag{3.345}$$

Для кодеру з параметрами $K = 4$, $\frac{1}{n} = \frac{1}{3}$, структурна схема якого наведена на рис. 3.96, три компоненти вектору вихідного сигналу **U** визначаються співвідношеннями:

$$u_1 = \mathrm{mod}_2(b_1, b_2, b_3, b_4); \qquad u_2 = \mathrm{mod}_2(b_1, b_2, b_3),$$
$$u_3 = \mathrm{mod}_2(b_2, b_3, b_4). \tag{3.346}$$

Аналогічно, для кодеру з параметрами $K = 5$, $\frac{1}{n} = \frac{1}{3}$, структурна схема якого наведена на рис. 3.69, можна записати наступні вирази для трьох компоненти вектору вихідного сигналу **U**:

$$u_1 = \mathrm{mod}_2(b_1, b_2, b_3, b_4, b_5); \qquad u_2 = \mathrm{mod}_2(b_1, b_2, b_3, b_4),$$
$$u_3 = \mathrm{mod}_2(b_2, b_3, b_4, b_5). \tag{3.347}$$

У загальному випадку кількість контрольних сум визначається параметром n, кількість бітів буферного регістру дорівнює K, а спосіб формування контрольних сум визначається схемою кодувального пристрою. Згідно з теоретичними відомостями, наведеними у підрозділі 3.8.2, кількість станів кодеру d_{st} становить

$$d_{st} = 2^{K-1}, \tag{3.348}$$

а поточний стан кодеру визначається вмістом перших лівих K – 1 бітів буферного регістру. Ці загальні правила теорії згорткових кодів також були використані для формування матриці станів, спосіб створення та загальні властивості якої були описані у підрозділі 3.8.12.1.

Узагальнений алгоритм формування згорткового коду за умови відомих значень параметрів кодеру та бітів вхідної послідовності **C** можна записати наступним чином.

1. Зчитати значення параметрів кодеру n та K та визначити за цими параметрами довжину буферу та кількість контрольних сум.

2. Спосіб обчислення контрольних сум задається автоматично згідно з обраною схемою кодеру.

3. Цілком зчитується вектор вхідної кодової послідовності **C** та визначається його довжина $l_{\mathbf{C}}$.

4. Встановлюється початковий стан кодеру, зазвичай він відповідає нульовим значенням всіх бітів буферного регістру.

5. Проводиться обробка вектору вхідної послідовності згідно з наступним ітераційним алгоритмом:

$$i = 1 \ldots l_{\mathbf{C}}; \quad j = 2 \ldots K; \quad j_{\text{вих}} = (n(i-1)+1) \ldots ni$$

$$b_j = b_{j-1}; \qquad b_1 = \mathrm{C}_i; \qquad \mathbf{U}_{j_{\text{вих}}} = F(\mathbf{B}), \tag{3.349}$$

де $i, j, j_{\text{вих}}$ – параметри ітерації, C_i – елементи вектору вхідної послідовності **C**, F(**B**) – векторна функція, яка формує біти вектору вихідної кодової послідовності **U** відповідно до значень бітів буферного регістру, заданого поточним значенням компонент вектору **B**. Загальні основи математичної теорії вектор-функцій та можливості їхнього використання в кодах комп'ютерних програм як ефективного засобу матричного програмування розглядалися в навчальних посібниках [13, 14]. Наприклад, загальному співвідношенню **U** = F(**B**) цілком відповідають наведені вище системи лінійних рівнянь двійкової арифметики (3.345), (3.346) та (3.347).

Блок-схема описаного вище узагальненого алгоритму формування згорткового коду наведена на рис. 3.98. Щодо описання векторної функції

F(**B**), воно проводиться окремо для кожного типу кодеру з використанням відповідних функцій двійкової арифметики. Як було відмічено у підрозділі 3.8.1, головною з цих функцій є функція сумування за модулем 2.

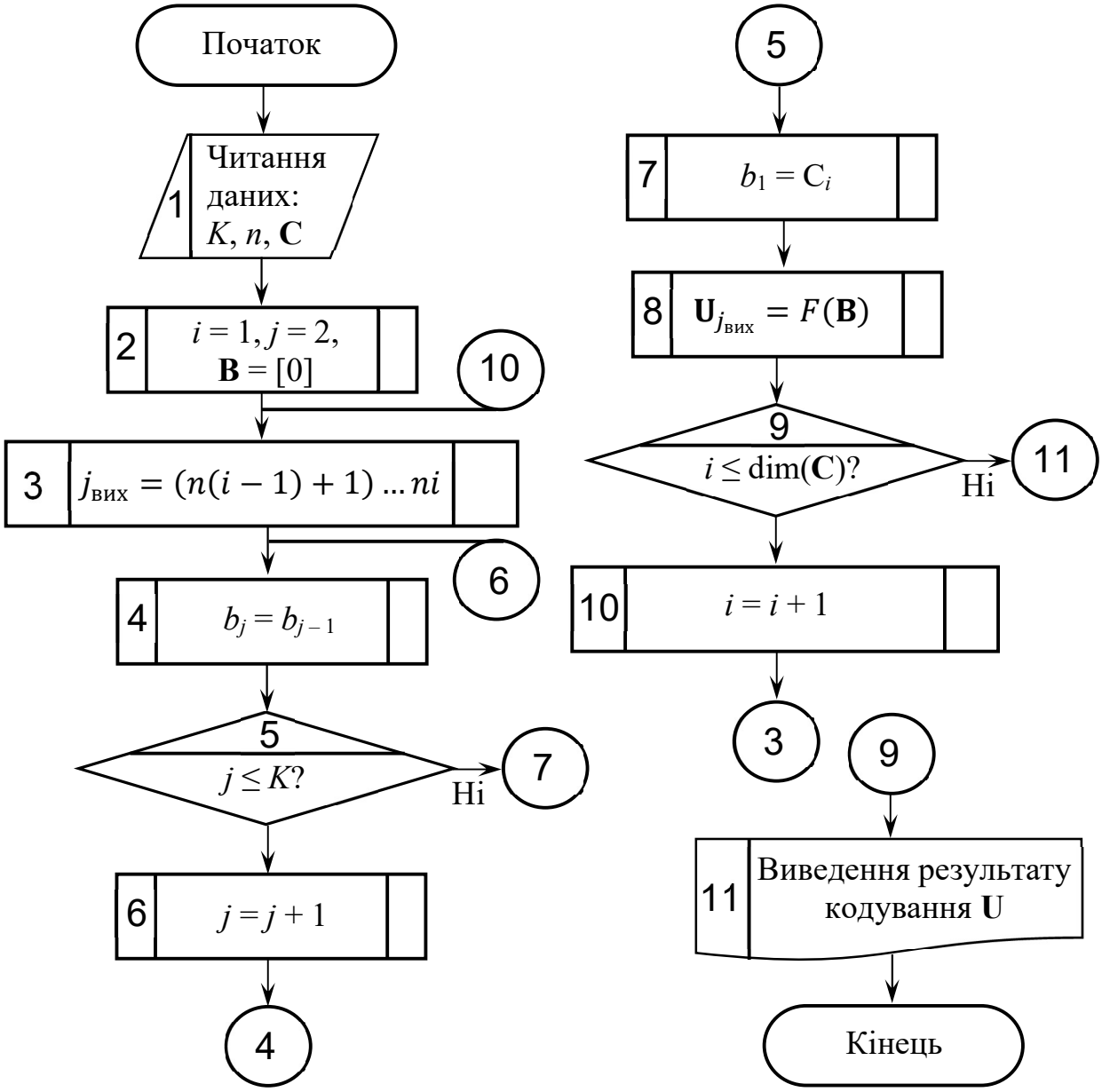


Рис. 3.98 Узагальнений алгоритм формування послідовності згорткового коду

Алгоритм формування згорткових кодів, описаний в цьому підрозділі, був реалізований в комп'ютерній програмі **`cnvcode`**, початковий код якої наведений у додатку П. Ця програма написана з використанням відомих

програмних засобів логічного, структурного, функціонального та матричного програмування системи науково-технічних розрахунків MatLab [13, 14]. Особливості написання коду цієї програми розглядатимуться в підрозділі 3.8.12.6.

3.8.12.4 Алгоритм пошуку шляху за ґратковою діаграмою згорткового коду через аналіз матриці станів

Зрозуміло, що алгоритми декодування послідовностей згорткових кодів є значно більш складними, ніж узагальнений алгоритм формування таких послідовностей, який був описаний у підрозділі 3.8.12.3. Будемо формувати алгоритм декодування на основі аналізу структури матриці станів, спосіб формування та головні властивості якої розглядалися в підрозділі 3.8.12.1.

Спочатку розглянемо алгоритм пошуку оптимального шляху за ґратковою діаграмою згорткового коду. Із загальних теоретичних відомостей, наведених у підрозділі 3.8.5, зрозуміло, що у разі відсутності помилок прийнята та очікувана послідовності цілком співпадають та метрика шляху за Хеммінгом між цими послідовностями дорівнює 0. За такої умови можна вважати, що прийнята кодова послідовність розшифрована правильно. Не складає серйозних труднощів також аналіз коду у випадку одиночних помилок, коли різниця між прийнятою та очікуваною кодовими комбінаціями на протязі всього шляху є незначною. Тоді помилка відразу виправляється та правильною вважається очікувана кодова комбінація. Значно більш складним є випадок із великою кількістю пакетних помилок, поняття про які було надано у підрозділі 3.8.7. Тут проблема полягає в тому, що декодер в місці помилки може обрати хибний шлях та в результаті сформувати очікувану комбінацію, яка в кінцевих розрядах буде значно відрізнятися від переданої послідовності.

Тому розглянемо спочатку алгоритм пошуку оптимального шляху за метрикою без використанням методів його оптимізації. Цей алгоритм

базується на використанні аналізу структури матриці станів згорткового коду. Метод виправлення пакетних помилок у згорткових кодах, оснований на повторному аналізі матриці станів з використанням алгоритму послідовного декодування, розглядатиметься у наступному підрозділі.

Формалізований алгоритм пошуку оптимального шляху за ґратковою діаграмою згорткового коду з використанням матриці станів можна записати наступним чином.

1. Зчитуються параметри коду. За параметром n визначається довжина кодового слова, за параметром K – кількість станів системи, а за структурною схемою коду – можливі переходи між його станами. На основі інформації про можливі переходи між станами записується матриця станів.

2. Зчитується прийнята кодова послідовність **C**. Якщо її довжина не є кратною параметру надлишковості коду n, така послідовність вважається помилковою.

3. На початку процесу декодування встановлюється номер ітерації $t = 1$ та початковий стан кодувальної системи $i = 1$.

4. Прийнята кодова послідовність розбивається на окремі слова $\mathbf{P}_s$ із довжиною n бітів. Відповідні аналітичні співвідношення можна записати анступним чином:

$$s = \dim(\mathbf{C})/n, \qquad \mathbf{P}_t = \mathbf{C}_{t(n-1)+1} \ldots \mathbf{C}_{tn}. \tag{3.350}$$

де s – кількість ітерацій процесу декодування. Тобто, $t = 1...s$.

5. Через аналіз відповідного стовпчику матриці станів визначаються можливі переходи із поточного стану i до інших станів системи j_0 та j_1 кодові послідовності $\mathbf{W}_0$ та $\mathbf{W}_1$, які відповідають цим переходам, та відповідне значення вихідного сигналу декодера $m = 0$ або $m = 1$. У введених позначеннях індекси 0 відповідають випадку $m = 0$, а індекси 1 – випадку $m = 1$.

6. Здійснюється порівняння можливих кодових послідовностей та прийнятої послідовності $\mathbf{P}_s$ за метрикою Хеммінга. Правильною

послідовністю **T** вважається та з можливих кодових послідовностей $\mathbf{W}_0$ та $\mathbf{W}_1$, для якої метрика за Хеммінгом, R_0 або R_1 є меншою, тобто:

$$R_0 = |\ \mathbf{W}_0 \oplus \mathbf{P}_s\ |,\ R_1 = |\ \mathbf{W}_1 \oplus \mathbf{P}_s\ |,\ \mathbf{T} = \begin{cases} \mathbf{W}_0, & \text{якщо } R_0 \le R_1; \\ \mathbf{W}_1, & \text{якщо } R_0 > R_1. \end{cases} \quad (3.351)$$

7. Згідно з проведеним аналізом матриці станів змінюється поточний стан системи $i \rightarrow j_0$ або $i \rightarrow j_1$ та формується елемент вектору вихідного цифрового сигналу декодеру **Q**. Зрозуміло, що наступний стан системи j_t та поточний елемент вихідного вектору $\mathbf{Q}_t$ визначаються наступним чином:

$$j_t = \begin{cases} j_0, & \text{якщо } R_0 \le R_1; \\ j_1, & \text{якщо } R_0 > R_1. \end{cases} \qquad m = \begin{cases} 0, & \text{якщо } R_0 \le R_1; \\ 1, & \text{якщо } R_0 > R_1. \end{cases}$$

$$\mathbf{Q}_t = m,\ i = j_t. \quad (3.352)$$

8. Якщо аналіз кодової послідовності не закінчений – аналізуються наступні n бітів кодової послідовності. Відповідно, номер ітерації t збільшується на 1, тобто $t = t + 1$, та здійснюється перехід до пункту 4 алгоритму. Нова кодова послідовність $\mathbf{P}_s$ формується на основі співвідношення (3.350).

У противному випадку – кінець аналізу прийнятої кодової послідовності та виведення результату декодування.

У описаному вище алгоритмі пошуку оптимального шляху для поточного стану декодеру окремим завданням є аналіз матриці станів **M**. Розглянемо тепер алгоритм аналізу матриці станів окремо. Цей алгоритм ґрунтується на загальних властивостях матриць станів, описаних у підрозділі 3.8.12.1.

1. Визначення за номером стану системи стовпчика матриці станів i, який необхідно проаналізувати.

2. Вважати, що номер рядка матриці j дорівнює 1.

3. Якщо елемент матриці, який аналізується, дорівнює –1, такий перехід між станами є неможливим.

4. Якщо елемент матриці станів, який аналізується, дорівнює 1,

встановити значення 1 для вихідного сигналу та прочитати можливе значення вхідного сигналу, аналізуючи стовпчик i та рядок j для інших сторінок матриці, від другої до останньої. Сформувати вектор можливої вхідної кодової послідовності.

5. Якщо елемент матриці станів, який аналізується, дорівнює 0, встановити значення 0 для вихідного сигналу та прочитати можливе значення вхідного сигналу, аналізуючи стовпчик i та рядок j для інших сторінки матриці, від другої до останньої. Сформувати вектор можливої вхідної кодової послідовності.

6. Якщо значення поточного рядка i відповідає номеру останнього можливого стану кодувальної системи d_{st}, який визначається співвідношенням (3.348), аналіз поточного стану вважається закінченим. У противному випадку – збільшення значення i на 1 та перехід до пункту 3.

Блок-схема алгоритму пошуку оптимального шляху за ґратковою діаграмою, сформованого на основі співвідношень (3.350) – (3.352), наведена на рис. 3.99, а блок-схема алгоритму аналізу стовпчика матриці станів – рис. 3.100. В блок-схемі алгоритму, яка наведена на рис. 3.99, замість розгалуження переходів на наступний стан декодеру через індекси j_0 та j_1, використаний уніфікований індекс j_m. Аналогічно проіндексовані очікувані кодові послідовності $\mathbf{W}_0$ та $\mathbf{W}_1$. Це у значній мірі спрощує наочне подання структури описаного алгоритму та дозволяє уникнути зайвих розгалужень.

Описаний алгоритм також реалізований у програмі **cnvcode**, початковий код якої наведений у додатку П. Описання структури згорткових кодів через матриці стану дозволило у значній мірі спростити програмний код та ефективно використовувати відомі засоби матричного програмування системи науково-технічних розрахунків MatLab [13, 14]. Особливості написаного програмного коду, а також структура програми та використані засоби програмування, будуть розглядатися в підрозділі 3.8.12.6. Перевага описаних алгоритмів аналізу станів декодера, пов’язаних з аналізом матриці станів, полягає саме у можливості використання для їхньої комп’ютерної реалізації ефективних сучаних методів матричного програмування [13, 14].

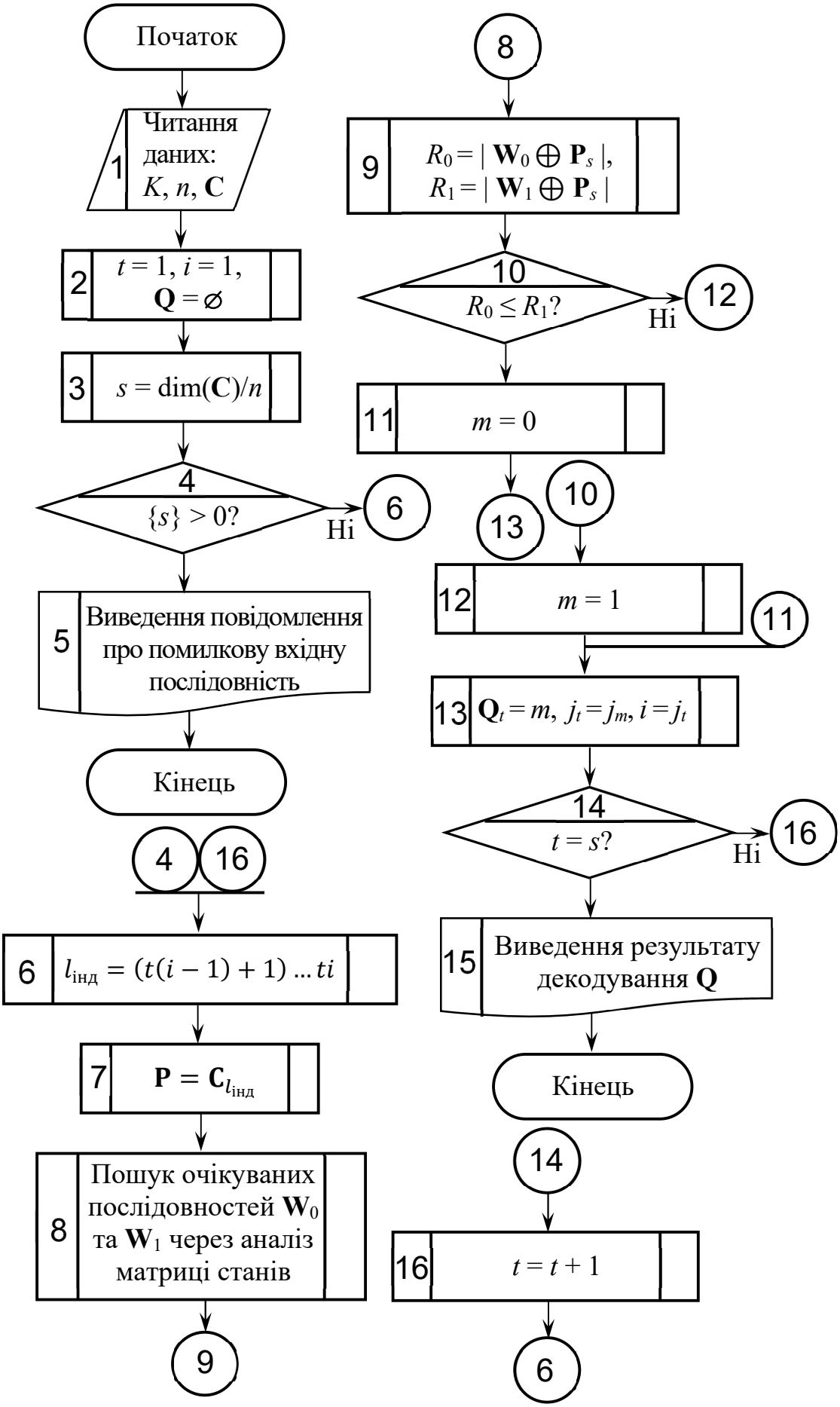


Рис 3.99 Блок-схема алгоритму пошуку шляху за мінімальною метрикою

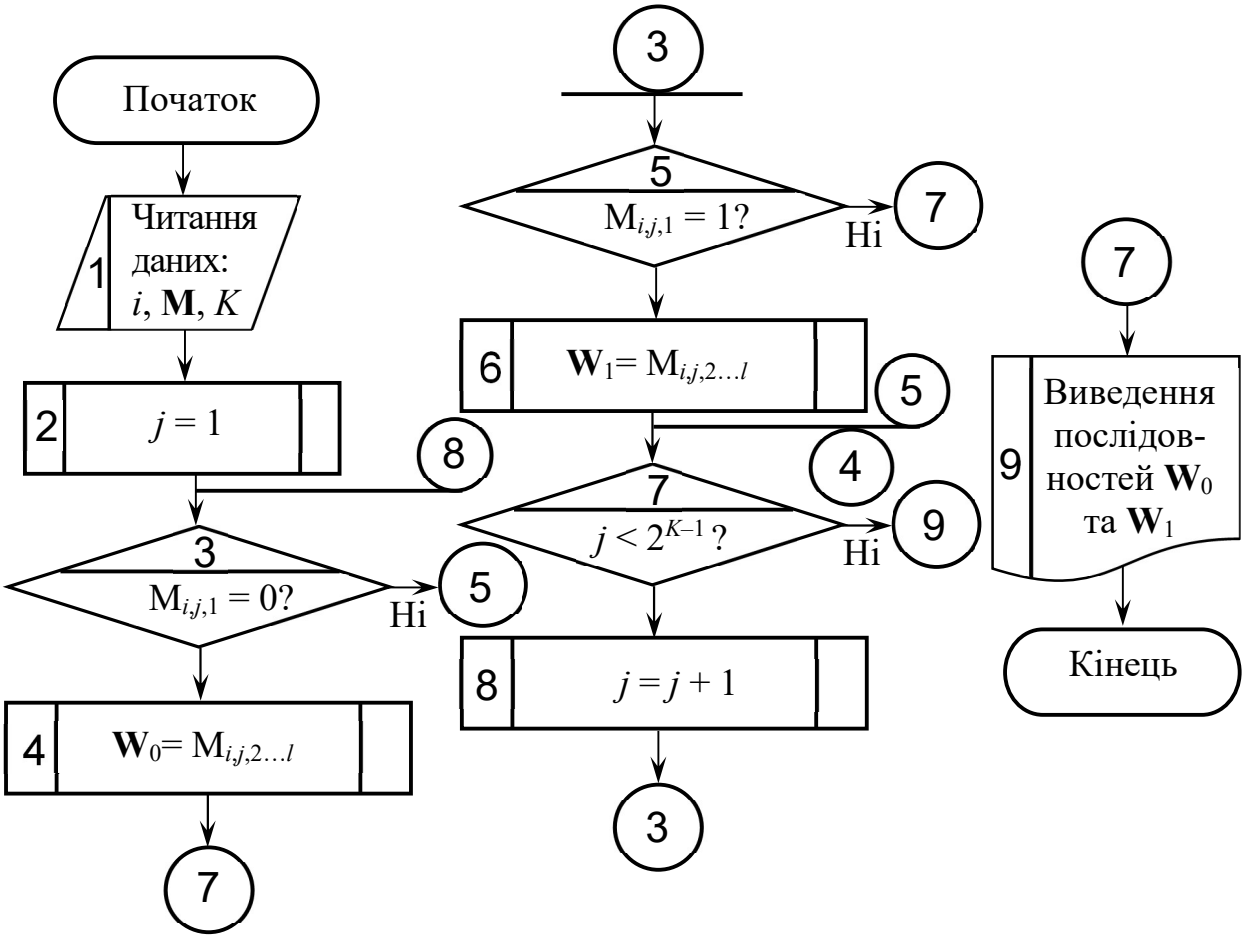


Рис 3.100 Узагальнена блок-схема алгоритму формування кодових послідовностей $\mathbf{W}_0$ та $\mathbf{W}_1$ через аналіз матриць стану

3.8.12.5 Алгоритм виправлення помилок у послідовностях згорткових кодів через пошук шляху із найменшою метрикою

Пошук пакетних помилок у згортковому коді через аналіз структури ґраткової діаграми та обчислення метрики різних шляхів є окремою важливою задачею теорії кодування. Без адаптації до аналізу пакетних помилок алгоритм пошуку мінімального шляху за поточним станом декодеру, який був розглянутий у попередньому підрозділі та ґрунтується на використанні матриці станів, може давати хибну кодову послідовність у разі їхнього виникнення. А саме пакетні помилки досить часто виникають в системах цифрового зв’язку за умови випадкової дії сильної електромагнітної завади та різкого зменшення співвідношення потужності корисного сигналу до

потужності шуму [1]. Така пакетна помилка може вивести декодувальну систему згорткового коду на хибний шлях, і за такої умови кодові слова починають розшифровуватися неправильно навіть після закінчення дії сигналу завади.

Тому розглянемо тепер, як можна пристосувати алгоритм пошуку оптимального шляху за поточним станом декодера, який ґрунтується на аналізі структури матриці станів та був описаний у підрозділі 3.8.12.4, до оптимізації знайденого шляху за ґратковою діаграмою у разі формування неправильної вихідної послідовності через виникнення пакетних помилок. Така оптимізація шляху проводиться за критерієм неменшої метрики Хеммінга вздовж ґратки з використанням алгоритму послідовного декодування, описаного в підрозділі 3.8.4.1.

Головною ознакою впливу пакетної помилки на порушення алгоритму роботи декодеру є те, що, починаючи з відповідного символу до кінця послідовності, метрики шляху за Хеммінгом є досить високими та не дорівнюють 0. Якщо в процесі аналізу структури ґратки за поточним станом через використання матриці станів формувати вектор станів системи, та вектор метрик шляху на поточному стані, можна, аналізуючи вектор метрик шляху, знайти той вузол ґратки, від якого почав формуватися ланцюжок помилок, що йдуть до кінця кодової комбінації. Це є ознакою наявності пакетної помилки в цій комбінації. У разі наявності такої помилки та сформованих векторів стану системи та метрик шляху існує можливість повернутися до вузла ґратки, в якому метрика за Хеммінгом вперше набуває ненульове значення та спробувати проаналізувати єдиний альтернативний шлях з цього вузла. Якщо метрика за Хеммінгом між отриманими та очікуваними послідовностями для альтернативного шляху є більшою, необхідно повернутися до попереднього варіанту та почати аналіз з наступного вузла, а якщо більшою – змінити шлях на альтернативний. Такий аналіз станів декодеру слід продовжувати, доки метрика всього шляху та стане меншою за критичне значення. Іншою ознакою виправлення пакетної помилки згорткового коду є нульові значення метрик

наприкінці кодової комбінації, але ця ознака має місце лише тоді, коли пакетна помилка виникає на початку або всередині кодової комбінації.

Для реалізації алгоритму пошуку пакетних помилок у послідовностях згогрткових кодів необхідно ввести два додаткових параметри. Першим з них є мінімальна припустима метрика розшифрованої кодової комбінації R_{cr}, за умови перевищення якої слід проводити повторний аналіз коду у зворотному напрямку з метою пошуку можливих шляхів із меншими метриками. А другим параметром є кількість нульових метрик $z_{\min}$ у сусідніх зліва вузлах ґратки. Якщо кількість вузлів, які стоять безпосередньо зліва від вузла з номером j_{cr} та мають нульову метрику, перевищує критичну величину $z_{\min}$, можна, із великою імовірністю, припустити, що виникла пакетна помилка, початком якої є саме вузол j_{cr}. Пошук критичної величини помилкових розрядів згорткового коду безпосередньо пов'язаний із імовірнісними оцінками, відповідні теоретичні відомості розглядалися у підрозділі 3.8.10.

У будь-якому разі зрозуміло, що імовірність спотворення великої кількості бітів є вкрай низкою, відповідні теоретичні оцінки з використанням схеми повторних випробувань Бернуллі були проведені у другому томі другої частини посібника [49]. Проте слід мати на увазі, що поява одного або двох нулів в метриках станів можлива і у спотвореній кодовій послідовності. Це обумовлено тим, що кількість станів скінченного автомату є обмеженою і в процесі аналізу кодової послідовності ці стани можуть повторюватись. Наприклад, розглянемо послідовність [0, 0, 0, 0, 0, 0, 2, 3, 2, 0, 0, 3, 3, 2, 1]. Якщо аналізувати її з кінця в лівому напрямку, можна побачити, що на десятій та одинадцятій позиції стоять два нулі. Але це не свідчить про те, що до дванадцятої позиції декодер опрацьовував код безпомилково, оскільки метрики станів, які стоять на позиціях 9, 8 та 7, також є ненульовими. А вже далі, починаючи з шостої позиції, стоять шість нулів. Тому пошук інших шляхів з меншою загальною метрикою кодової комбінації слід починати з сьомого розряду.

Розглянемо покроковий алгоритм оптимізації шляху за ґратковою

діаграмою за умови відомих векторів станів системи **S** та метрик шляху в кожному вузлі ґратки **Rn**.

1. Зчитування вхідної кодової комбінації **C**, параметрів коду K та n, а також розглянутих вище критичних параметрів R_{cr} та $z_{\min}$.

2. З використанням алгоритму пошуку мінімального шляху поточного стану (АПМШПС) проводиться аналіз вхідної кодової комбінації. Цей алгоритм, оснований на аналізі структури матриці станів, був розглянутий у підрозділі 3.8.12.4. Відмінність використання АПМШПС в алгоритмі оптимізації шляху за структурою ґратки, який розглядається, від описаної у підрозділі 3.8.12.4 базової версії АПМШПС полягає у тому, що в процесі проходження ґратки з використанням АПМШПС формуються вектори **S** та **Rn**.

3. За умови відомих значень вектору **Rn** через сумування його компонентів обчислюється загальна метрика шляху $\mathbf{R}_{\Sigma_1}$.

4. Якщо $\mathbf{R}_{\Sigma_1} \le R_{cr}$ – отримана декодована послідовність **Q** вважається правильною та обчислювальний процес завершується. У противному випадку здійснюється перехід до п’ятого пункту алгоритму.

5. Аналіз вектору **Rn** для отриманої вихідної послідовності **Q** та пошук критичного стану системи j_{cr} за наступним логічним співвідношенням теорії предикатів:

$$\exists(j_{cr} > z_{\min})\left((\forall(i < j_{cr}), \mathbf{Rn_i} = 0)\&\left((i \ge j_{cr}), \exists(\mathbf{Rn_i} \ne 0)\right)\right). \quad (3.353)$$

6. Читання з вектору станів системи **S** елементів $\mathbf{S}_{j_{cr}}$ та $\mathbf{S}_{j_{cr}-1}$ та зміна поточного стану декодеру через аналіз стовпчика з номером $j_{cr}-1$ матриці станів наступним чином:

$$\mathbf{S}_{j_{cr}} = p, \mathbf{S}_{j_{cr}} = \begin{cases} \left(\mathbf{M}_{\mathbf{S}_{j_{cr}-1}, j_0, 2\ldots l}\right)_{10}, & \text{якщо } p = j_1; \\ \left(\mathbf{M}_{\mathbf{S}_{j_{cr}-1}, j_1, 2\ldots l}\right)_{10}, & \text{якщо } p = j_0. \end{cases} \quad (3.354)$$

7. Зміна значення поточного біту вектору двійкової вихідної послідовності $\mathbf{Q}_{j_{cr}}$ на інверсне:

$$\mathbf{Q}_{j_{cr}} = \overline{\mathbf{Q}_{j_{cr}}}. \quad (3.355)$$

8. Подальший послідовний аналіз зміни станів декодеру для номерів стану $j_{cr+1} \le j \le j_{\max}$ за алгоритмом АПМШПС.

9. Повернення до пункту 4 алгоритму.

Блок-схема описаного алгоритму виправлення помилок наведена на рис. 3.101.

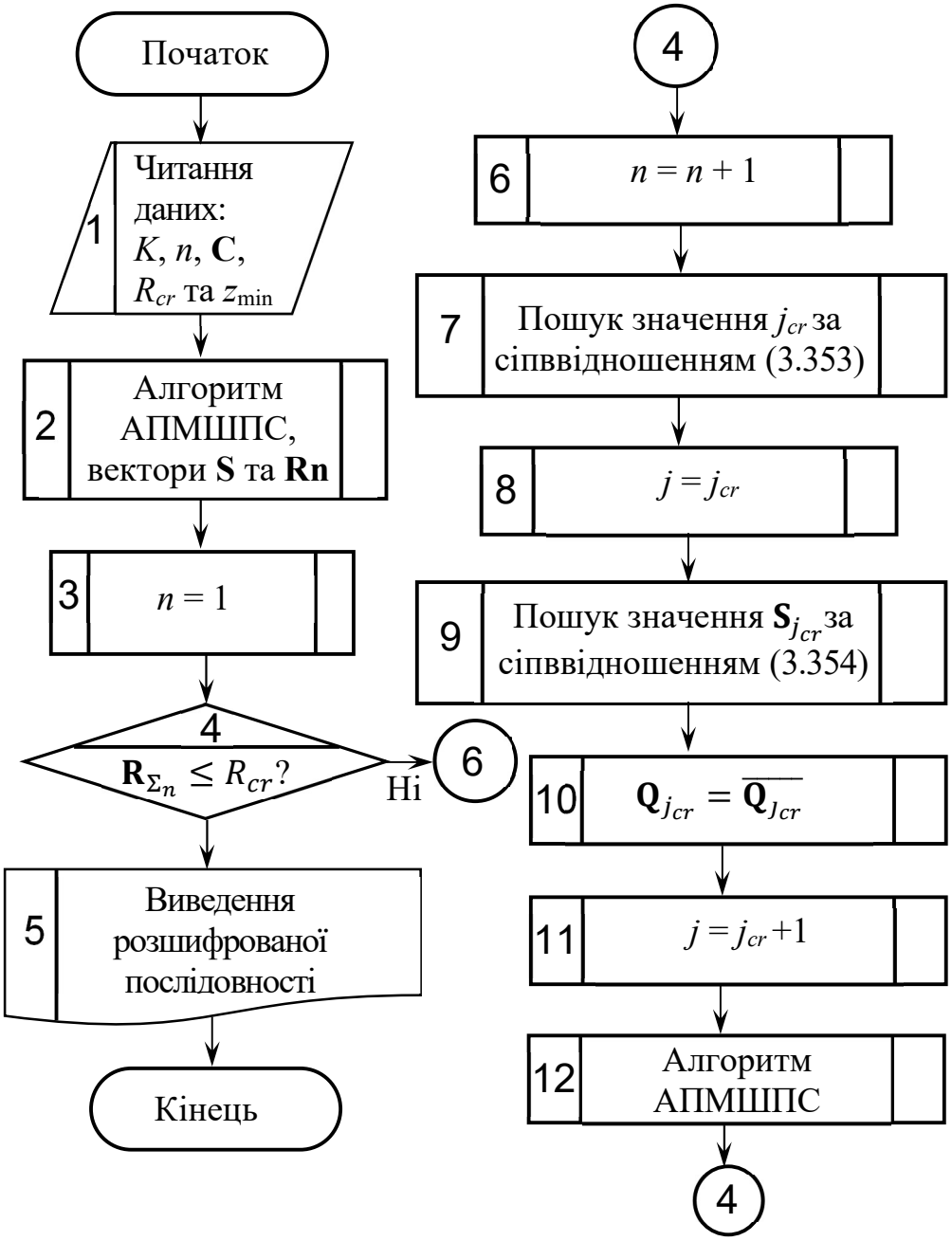


Рис. 3.101 Алгоритм пошуку пакетних помилок у прийнятій комбінації згорткового коду, в основу якого покладений алгоритм АПМШПС

3.8.12.6 Особливості написання комп'ютерної програми, призначеної для формування та розшифрування послідовностей згорткових кодів за описаними алгоритмами

Код програми **cnvcode**, призначеної для формування послідовностей завадостійкого коду та їхнього декодування, написаний мовою програмування системи науково-технічних розрахунків MatLab [13, 14], наведений у додатку П. Операції кодування та декодування виконуються для послідовностей згорткових кодів із параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$; $K = 4$, $\frac{1}{n} = \frac{1}{3}$ та $K = 5$, $\frac{1}{n} = \frac{1}{3}$, відповідні структурні схеми кодерів та матриці станів, які їх відповідають, розглядалися в підрозділах 3.8.3, 3.8.5 та 3.8.12.2.

Для формування послідовностей згорткових кодів реалізований алгоритм формування контрольних сум, описаний у підрозділі 3.8.12.3, а для декодування цих послідовностей – алгоритм АПМШПС, розглянутий у підрозділі 3.8.12.4, та алгоритм пошуку пакетних помилок, розглянутий у підрозділі 3.8.12.5.

Розглянемо послідовно особливості написання коду програми, наведеної у додатку П.

Як вхідні параметри цієї програми використовують наступні структури та числові дані.

1. Вектор вхідної послідовності **vin**, яку необхідно закодувати або декодувати.

2. Змінна **notsk**, яка визначає номер завдання. Ця змінна є цілою величиною і може приймати лише 2 значення: **notsk=1** для завдання кодування та **notsk=2** для завдання декодування.

3. Змінна **kp**, яка визначає тип коду. Ця змінна також є цілою величиною і може приймати 3 наступних значення:

kp = 1 – для коду з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$;

kp = 2 – для коду з параметрами $K = 4$, $\frac{1}{n} = \frac{1}{3}$;

kp = 3 – для коду з параметрами $K = 5$, $\frac{1}{n} = \frac{1}{3}$.

В результаті роботи програми формується вихідний вектор **vout**, яким є послідовність згорткового коду у разі розв'язування завдання кодування або початкова двійкова послідовність у разі розв'язування завдання декодування.

На початку програми здійснюється контроль вхідних параметрів за наступними ознаками.

1. Всі елементи вектору вхідної двійкової послідовності, як для завдання кодування, так і для завдання декодування, повинні дорівнювати 0 або 1.

2. Значення вхідного параметру **notsk**, згідно з наведеним вище описанням призначення цієї змінної, має бути 1 або 2.

3. Значення вхідного параметру **kp**, згідно з наведеним вище описанням призначення цієї змінної, має бути 1, 2 або 3.

4. Для завдання формування кодової послідовності (**notsk=1**) кількість елементів **nv** вхідного вектору **vin** не має перевищувати значення 20.

5. Для завдання декодування кодової послідовності (**notsk=2**) кількість елементів **nv** вхідного вектору **vin** не має перевищувати значення 60.

Далі в програмі проводиться зміна порядку слідування бітів числа із прямого до зворотного. Це пов'язано з тим, що читати двійкові послідовності зручно у прямому порядку слідування бітів, зправо-наліво, а їх обробка в програмі здійснюється у зворотному порядку слідування, зліва-направо. У тексті цього посібника не один раз підкреслювалось, що для правильної роботи будь-яких засобів кодування та декодування обраний порядок слідування бітів числової послідовності має вельми важливе значення. Для зміну порядку слідування бітів двійкової послідовності у програмі використані наступні рядки команди циклу **for** системи науково-технічних розрахунків MatLab [13, 14]:

```
for iii=1:nv
    vin_inv(iii)=vin(nv-iii+1);
end; %for iii=1:nv
```

Для розв’язування завдання формування кодової послідовності використані наступні змінні.

Vout – вектор вихідної кодової послідовності, який на початку розв’язування завдання декодування є порожнім.

BUF – вектор, який описує буфер кодувального пристрою, який на початку розв’язування завдання декодування заповнюється нульовими елементами.

Для обчислення результатів контрольних сум згорткового коду використаний окремий програмний модуль **parity**, код якого також наведений у додатку П.

Програмний модуль **parity** працює за наступним алгоритмом.

1. На початку роботи програми значення змінної **n**, яка є результатом сумування за модулем 2 всіх елементів вхідного вектору, дорівнює 0.

2. Визначається довжина **nv** вхідного вектору **vin**.

3. Встановлюється початковий номер ітерації **i** для сумування елементів вхідного вектору за модулем 2.

4. Аналізується кожний елемент вхідного вектору **vin(i)**. Якщо **vin(i)=1**, аналізується поточне значення змінної **n**: за умови **n=0** вважається **n=1**, а у разі **n=1** вважається **n=0**. Інакше кажучи, якщо **vin(i)=1** поточне значення **n** змінюється на протилежне, а у разі **vin(i)=0** поточне значення **n** зберігається. За умови **vin(i)≠1** або **vin(i)≠0** виводиться повідомлення про помилку вхідного вектору даних та робота програми закінчується.

5. За умови **i<nv** вважається **i=i+1** та здійснюється перехід до наступної ітерації за пунктом алгоритму 4.

6. За умови **i=nv** виводиться обчислене значення **n** та робота програми закінчується.

Блок-схема описаного алгоритму наведена на рис. 3.102.

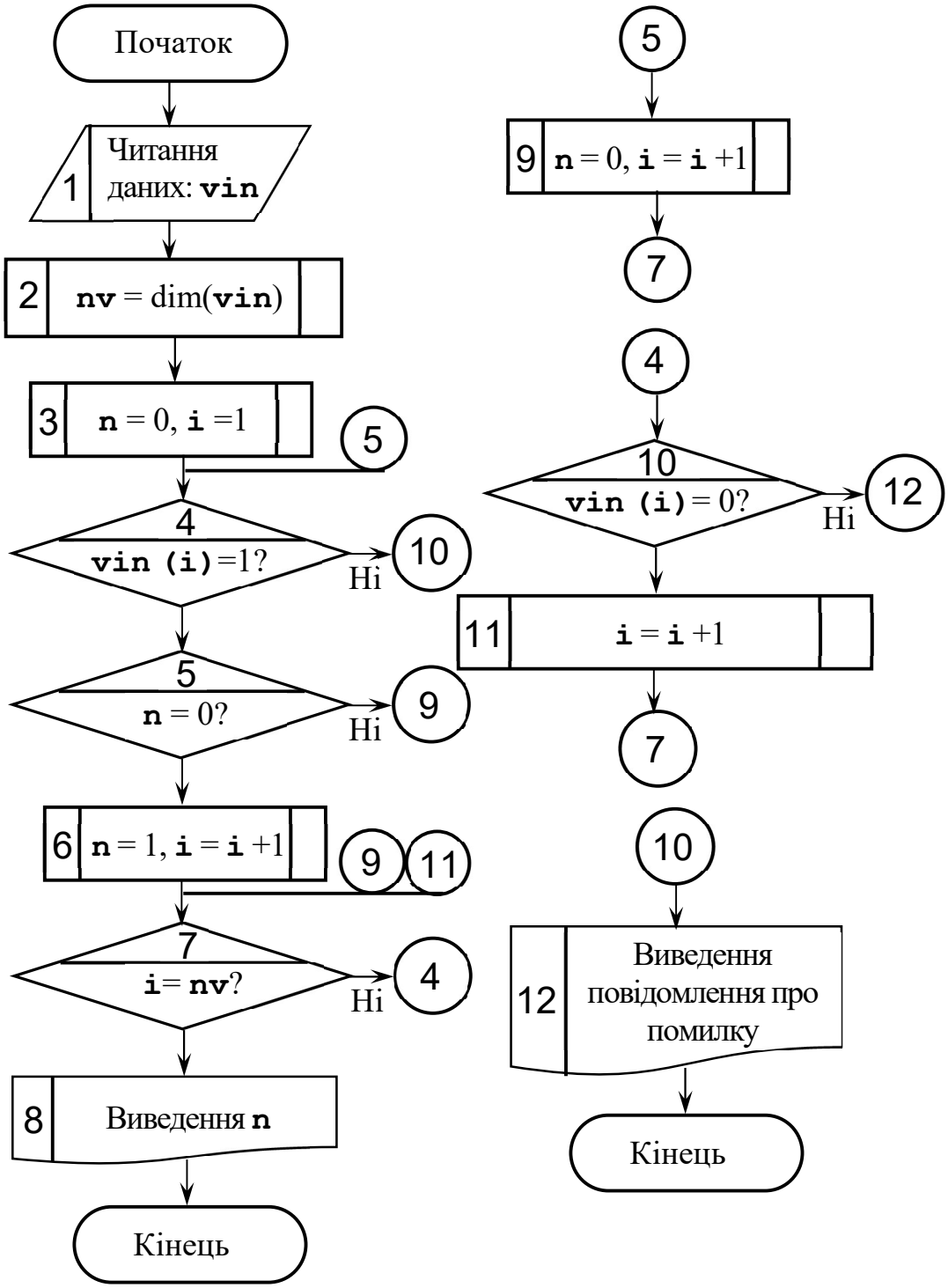


Рис. 3.102 Блок-схема алгоритму роботи програмного модуля **parity**, код якого наведений у додатку П

У модулі **parity** описаний алгоритм, блок-схема якого анведена на рис. 3.106, реалізований з використанням наступних програмних рядків мови програмування системи MatLab:

```
for ii=1:nv
```

```
    if((vin(ii)<0)|(vin(ii)>1)) error (ddwtr); end;
end;
for iii=1:nv
    if(vin(iii)==1)
        if (n==1) n=0; else n=1; end; %if(n==1)
  end;
```

Контрольні суми для формування послідовностей згорткового коду з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$ задаються через співвідношення (3.345), для коду з параметрами $K = 4$, $\frac{1}{n} = \frac{1}{3}$ – через співвідношення (3.346), та для коду з параметрами $K = 5$, $\frac{1}{n} = \frac{1}{3}$, – через співвідношення (3.347). Під час використання співвідношень (3.345) – (3.347) для роботи з буфером кодувального пристрою, визначеним через вектор **BUF**, використовуються наступні командні рядки мови програмування системи MatLab:

```
for iform=1:nv
    To_Buf=vin_inv(iform);
    Buf2=[To_Buf,BUF]; BUF=Buf2;
    BUF(4)=[];
    ..........................
 end;
```

Слід відзначити, що написаний програмний код можна легко удосконалити, надавши користувачам можливість формувати кодові послідовності для згорткових кодів з іншими параметрами. Для цього слід виконати наступні дії.

1. Розширити можливий діапазон значень змінної **kp**.

2. Встановити відповідну довжину буфера кодувального пристрою, яка визначається як K –1.

3. Сформувати контрольні суми, які у загальному вигляді задаються через універсальне рівняння (3.349).

4. Для зручності обчислення контрольних сум використовувати модуль **parity**, особливості написання початкового коду якого були розглянуті вище.

Головною особливістю чистини програмного коду модуля **cnvcode**, яка використовується для аналізу та декодування кодових послідовностей згорткових кодів, є використання матриць стану, які описуються безпосередньо у коді програми через формування тривимірного масиву даних з іменем **Mcorr** та надання його змінним відповідних значень. Матриці стану для кодів із заданими параметрами, як і в прикладах 3.66 – 3.68, описуються посторінково через формування двовимірних масивів, а їх об'єднання до тривимірних масивів здійснюється з використанням відомого способу матричного програмування, який називається множиною індексацією елементів [13, 14]. Наприклад, для матриці, яка описую код з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$ та структура якої наведена у таблиці 3.30, відповідні командні рядки, написані з використанням мови програмування системи MatLab, мають наступний вигляд:

```
Mcorr1=[0,-1,0,-1; 1,-1,1,-1; ...
          -1,0,-1,0; -1,1,-1,1];
Mcorr2=[0,-1,1,-1; 1,-1,0,-1; ...
         -1,1,-1,0; -1,0,-1,1];
Mcorr3=[0,-1,1,-1; 1,-1,0,-1; ...
         -1,0,-1,1; -1,1,-1,0];
Mcorr=Mcorr1; Mcorr(:,:,2)=Mcorr2;
Mcorr(:,:,3)=Mcorr3;
```

Аналогічно записуються матриці для кодів з параметрами $K = 4$, $\frac{1}{n} = \frac{1}{3}$ та $K = 5$, $\frac{1}{n} = \frac{1}{3}$. Посторінкова структура матриці станів для коду $K = 4$, $\frac{1}{n} = \frac{1}{3}$ представлена в таблиці 3.32, а для коду $K = 5$, $\frac{1}{n} = \frac{1}{3}$ – на рис. 3.95.

Іншою відмінною рисою програмного коду, в якому реалізований алгоритм АПМШПС, описаний у підрозділі 3.8.12.4, є універсальні засоби реалізації для аналізу структури матриць стану будь-якого порядку. Для проведення аналізу матриці станів через програмний код в автоматичному режимі, незалежно від типу згорткового коду, в програму введені наступні три змінні.

1. **NST** – кількість станів системи. Згідно із властивістю 3.24 кількість станів кодувальної системи $s = 2^{K-1}$. Тобто, для згорткових кодів, які розглядаються, ця змінна має наступні значення.

Для коду з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$ – **NST=4**.

Для коду з параметрами $K = 4$, $\frac{1}{n} = \frac{1}{3}$ – **NST=8**.

Для коду з параметрами $K = 5$, $\frac{1}{n} = \frac{1}{3}$ – **NST=16**.

2. **Nsimb** – кількість символів коду на 1 вхідний біт. Згідно із властивістю 3.24 цей параметр матриці стану відповідає параметру надлишковості коду $\frac{1}{n}$. Тобто, за умови $\left(\frac{1}{n} = \frac{1}{2}\right)$ **Nsimb=2**, а у разі $\left(\frac{1}{n} = \frac{1}{3}\right)$ – **Nsimb=3**.

3. **SUMMET** – додатковий параметр, який відповідає сумі елементів матриці станів для випадку, коли перехід з поточного стану до стану із заданим номером є неможливим. Згідно із теоретичними відомостями, розглянутими в підрозділі 3.8.12.1, за такої умови елементи всіх сторінок матриці станів дорівнюють –1. Значення змінної **SUMMET** формується як сума всіх елементів із заданим номером рядка та стовпчика, з другої до останньої сторінки матриці. Тобто, кількість елементів, які сумуються, відповідає параметру надлишковості згорткового коду n. Відповідно для коду з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$, **SUMMET=-2**, а для кодів з параметрами $K = 4$, $\frac{1}{n} = \frac{1}{3}$ та $K = 5$, $\frac{1}{n} = \frac{1}{3}$ **SUMMET=-3**.

Для зміни поточного стану системи на наступний також використовується проміжна змінна **STNew**, до якої записується номер стану, якому відповідає шлях з найменшою метрикою згідно з алгоритмом АПМШПС. На початок розшифрування кодової послідовності вводиться початкове значення цієї змінної **STNew=1**. З використанням цієї змінної формується масив станів **ST**, до яких можливий перехід із поточного стану. Саме за значеннями номерів станів, записаними до цього масиву, визначається шлях із поточного стану з мінімальною метрикою. Для визначення значення метрики шляху поточного стану використовується змінна **MET**.

Відповідний код програми, в якому реалізований алгоритм АПМШПС, має наступний вигляд.

```
for nstr=1:NST
        SIGN=Mcorr(nstr,ST(ifd),...
                2:Nsimb+1);
        SIGN_CMP=reshape(SIGN,1,...
                Nsimb,1);
      MET=sum(SIGN);
      if (MET~=SUMMET)
          CompM(nstr)=sum(xor...
              (SIGN_CMP,csec));
          if(CompMET(ifd)>...
              CompM(nstr))
              CompMET(ifd)=...
                  CompM(nstr);
            STNew=nstr;
         end;%if(CompMET(ifd)>
      end;
  end;
  MET_RES=sum(CompMET);
```

```
            Vout=[Vout,Mcorr(STNew,ST(ifd),1)];
        end;
```

В цьому коді, крім команди циклу із заданою кількістю повторень, використані відомі функції системи MatLab, призначені для роботи з матрицями та метод множинної індексації елементів матриці [13, 14].

Для реалізації алгоритму пошуку пакетних помилок використана додаткова змінна **Pos_Chng**, яка визначає передбачуваний номер такту роботи декодеру, в якому виникла пакетна помилка. Пошук відповідного такту здійснюється через масив **CompMET**, який являє собою пам'ять метрик шляхів, обраних декодером, для кожного такту. Пошук пакетних помилок згорткового коду здійснюється відповідно до алгоритму, який був описаний у підрозділі 3.8.12.5 з використанням оператору циклу з виходом за відповідною умовою **while**.

Результати роботи програмного модуля **convcode** для різних кодових послідовностей згорткового коду також наведені у додатку. Зрозуміло, що найбільш цікавими з них є результати декодування послідовностей із помилковими бітами.

3.8.13 Узагальнені оцінки коректувальних параметрів згорткових кодів

Перед вивченням цього підрозділу необхідно повторити підрозділи 2.2.2 та 3.8.4, а також підрозділи 3.8 та 7.3.2 другої частини посібника

3.8.13.1 Поняття про мінімальний просвіт згорткового коду

Розглянемо тепер просторові характеристики згорткових кодів з точки зору можливості виправлення помилок у спотворених кодових комбінаціях. Для цього необхідно повернутися до загальних відомостей теорії завадостійкого кодування, які були наведені у підрозділі 2.2.2 цієї частини посібника. Згідно з теорією завадостійкого кодування для згорткових кодів, як і для блокових та групових, нас буде цікавити мінімальна відстань між усіма

парами слів у коді, оскільки саме цей параметр характеризує коректувальну здатність коду.

Оскільки, згідно з загальною теорією завадостійкого кодування, згортковий код є лінійним груповим кодом, можна скористатися теоремою про те, що мінімальну кодову відстань можна шукати для будь-якої кодової послідовності, навіть для самої простої. Відповідні теоретичні відомості та теореми теорії кодування наведені у навчальній літературі [52, 53, 55, 62, 63]. Загалом це положення теорії завадостійкого кодування ґрунтується на тому, що для лінійних кодів всі тестові контрольні повідомлення є однаковими, незалежно від імовірності появи такого повідомлення та від його складності з точки зору кількості чергувань послідовностей з нулів та одиниць [33]. Тому будемо аналізувати коректувальну здатність згорткового коду, розглядаючи просту нульову вхідну послідовність. Загалом такий аналіз базується на обчислені мінімальної метрики серед всіх шляхів за ґратковою діаграмою, які можна вважати хибними відносно нульового шляху. Інакше кажучи, вважається, що у разі передавання нульової кодової послідовності хибною є будь-яка послідовність, що не є нульовою [33]. Помилка, про яку йде мова, полягає в тому, що відповідний шлях спочатку відхиляється від нульової послідовності, а потім знову до неї повертається. Тобто, будемо вважати, що помилка відбувається у будь-якому випадку, коли здійснюється відхилення від нульового шляху, або коли нульовий шлях «не виживає».

Може виникнути доречне питання: «Навіщо шукати на ґратковій діаграмі такі шляхи, які розбігається, а потім збігаються? Чи не достатньо і не варто було б шукати лише такі шляхи, які розбігається?». Дійсно, насправді, умова розбіжності шляхів є достатньою для обґрунтування висновку про те, що прийнята комбінація згорткового коду є спотвореною. Наприклад, у підрозділі 3.8.12.6 був розглянутий ітераційний алгоритм пошуку правильної кодової комбінації для шляху, який розбігається і потім не збігається, через аналіз матриці станів кодеру з використанням алгоритму послідовного

декодування. Проте, як було показано у підрозділі 3.8.12.6, у випадку, коли шляхи на ґратковій діаграмі повністю розбігаються, декодер, починаючи з цієї точки розбіжності, створює хибні послідовності. Такі помилки у теорії згорткових кодів називаються складними та розглядаються окремо [33, 52, 53, 55, 62, 63]. А тепер нас будуть цікавити помилки іншого типу, які, у разі їхньої появи, відразу виправляються декодером. Такий тип помилок розглядався у підрозділі 3.8.12.5, у теорії згорткових кодів такі помилки називаються простими [33, 52, 53, 55, 62, 63].

Розглянемо простий цифровий пристрій для формування згорткового коду з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$, структурна схема якого наведена на рис. 3.63, а ґраткова діаграма – на рис. 3.66. Для пошуку шляху з найменшою метрикою перерисуємо рис. 3.66, позначивши, як вагу кожної з гілок діаграми, не кодову послідовність, яка генерується, а кодову відстань за Хеммінгом між цією послідовністю та послідовністю 00. Така діаграма буде відрізнятися від аналогічної діаграми, наведеної на рис. 3.67, тим, що в діаграмі, наведеній на рис. 3.67, показані метрики гілок, обчислені між очікуваною та прийнятою кодовою послідовностями, а зараз нас будуть цікавити їхні метрики відносно нульової кодової послідовності. У разі такої зміни умови обчислення метрик ґраткова діаграма буде мати вигляд, показаний на рис. 3.103.

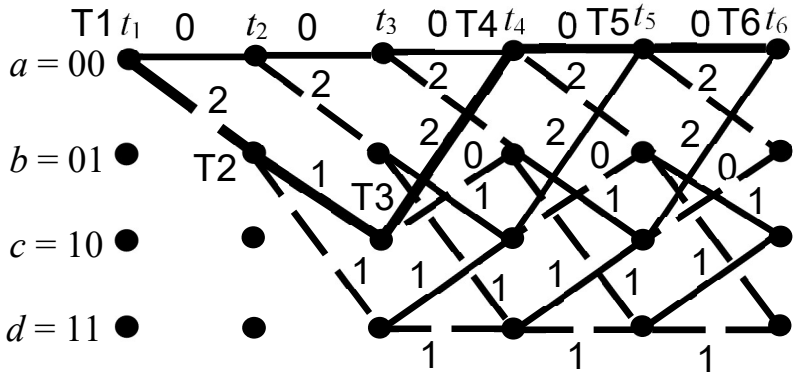


Рис. 3.103 Метрики гілок згорткового коду з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$ відносно кодової послідовності 00

Аналіз можливих шляхів від точки T1 до точки T6 показує, що найменшу метрику з них відносно послідовності 00 має шлях T1 → T2 → T3 → T4 → T5 → T6, показаний на рис. 3.103 жирною лінією, і метрика цього шляху дорівнює 5. Оскільки, як і раніше, на наведеній ґратковій діаграмі суцільні лінії відповідають біту нуля, а пунктирні лінії – біту 1, можна зробити висновок, що шляху із найменшою метрикою відносно нульової послідовності відповідає вхідна послідовність 100.

Надамо відповідні визначення, які є базовими для подальших міркувань щодо аналізу коректувальної здатності згорткових кодів.

Визначення 3.25. Мінімальна метрика шляху, який спочатку розбігається відносно нульової кодової комбінації, а потім знову її повторює, називається мінімальним просвітом (англійський термін – minimum free distance), або просто просвітом (англійський термін – free distance) [33, 52, 53, 55, 62, 63].

Зазвичай у технічній літературі мінімальний просвіт позначається символом d_f, і в теорії згорткових кодів цей параметр є аналогом мінімальної кодової відстані $d_{\min}$ для лінійних та блокових кодів [33, 52, 53, 55, 62, 63]. Згідно із співвідношенням (2.36), наведеним у підрозділі 2.2.2 цієї частини посібника, співвідношення для коректувальної здатності згорткового коду t можна записати наступним чином:

$$t = \left[\frac{d_f - 1}{2}\right]. \tag{3.356}$$

Зрозуміло, що наведене співвідношення (3.356) аналогічно співвідношенню (3.201), записаному для блокових кодів Ріда – Соломона.

Ґраткова діаграма, наведена на рис. 3.103, є наочною з тієї точки зору, що на ній показані всі шляхи, за якими можуть виникати помилки, та серед цих шляхів обраний один із найменшою метрикою. Якщо в програмований декодер на апаратному рівні закладено інформацію про можливі помилкові шляхи, він може через пошук шляху із мінімальним співвідношенням потужності сигналу до потужності шуму працювати більш стабільно.

3.8.13.2 Обчислення мінімального просвіту з використанням теорії скінченних автоматів

Спосіб обчислення мінімального просвіту через аналіз структури ґраткової діаграми згорткового коду, незважаючи на його наочність, також має один суттєвий недолік, який полягає у тому, що такий спосіб складно автоматизувати на алгоритмічному рівні та реалізувати у вигляді програми. Тому в теорії згорткових кодів розроблений інший спосіб обчислення мінімального просвіту, оснований на теорії скінченних автоматів [50]. Сутність цього способу полягає в тому, що через аналіз діаграми станів скінченного автомату формується аналітичний вираз для передаточної функції згорткового коду. Розглянемо цей спосіб для згорткового коду з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$, структурна схема якого наведена на рис. 3.63, а схема скінченного автомату, яка відповідає цьому кодувальному пристрою – на рис. 3.64. Перерисуємо схему скінченного автомату, наведену на рис. 3.64, наступним чином. Крім кодових комбінацій, які відповідають переходам з одного стану системи до іншого, будемо вказувати також метрики відповідних гілок. Спосіб формування таких метрик є наступним. Параметр кодової відстані між комбінацією, яка відповідає заданій гілці схеми скінченного автомату та нульовою кодовою комбінацією, позначається літерою D. Якщо прийнята та нульова комбінація не відрізняються, тобто гілці, яка розглядається, відповідає комбінація 00, вага цієї гілки складає $D^0 = 1$. Якщо прийнята та нульова комбінація різняться на 1 символ, тобто гілці, яка розглядається, відповідає комбінація 01 або 10, вага такої гілки складає $D^1 = D$. А якщо гілці відповідає кодова комбінація 11, вага такої гілки складає D^2. Відповідна схема скінченного автомату із позначеними ваговими коефіцієнтами гілок наведена на рис. 3.104.

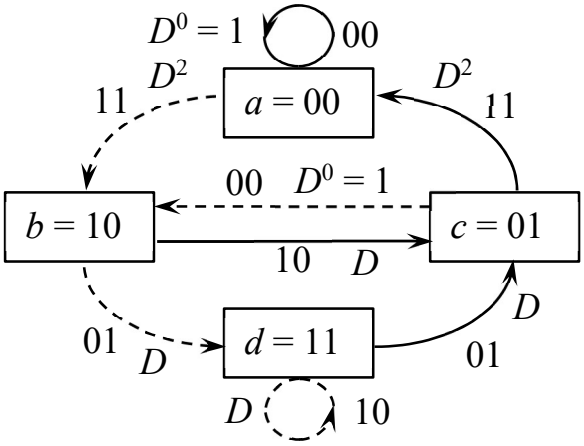


Рис. 3.104 Схема скінченного автомату для кодеру згорткового коду з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$ із вказаними метриками гілок відносно нульової кодової комбінації

Діаграма станів, наведена на рис. 3.104 має петлеві з'єднання, і тому вона є незручною для формування передавальної функції кодувальної системи та її запису у вигляді системи алгебраїчних рівнянь. Тому проведемо еквівалентне перетворення схеми скінченного автомату [50], наведеної на рис. 3.104, наступним чином [33].

1. Видалимо петлю *a*, яка ніяким чином не впливає на роботу скінченного автомату.

2. Проведемо розбиття вузла *a* на два вузла, *a* та *e*, таким способом, що вузол *a* буде вхідним, а *e* – вихідним вузлом схеми.

Відповідна перетворена схема скінченного автомату наведена на рис. 3.105.

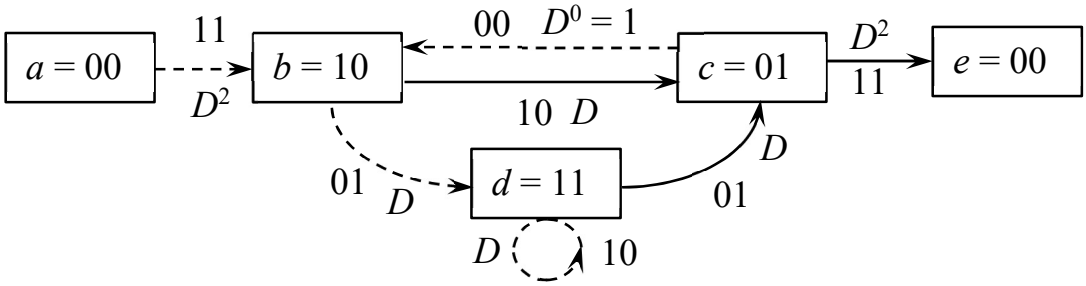


Рис. 3.105 Перетворена схема скінченного автомату, наведеного на рис. 3.104

Схему скінченного автомату, відображену на рис. 3.105, можна описати у вигляді системи лінійних рівнянь, сформованої за наступними правилами [33, 52, 53, 55, 62, 63].

1. Для формування рівняння для станів b, c, d та e необхідно просумувати вагу всіх гілок, які ведуть до відповідного стану.

2. Стани системи позначаються як X_a, X_b, X_c, X_d та X_e.

3. Коли гілки сумуються, кожна з них множиться на відповідну метрику.

З урахуванням сформульованих правил, запишемо систему рівнянь, яка описує роботу скінченного автомату, схему якого наведено на рис. 3.105, наступним чином [33]:

$$\begin{cases} X_b = D^2 X_a + X_c; \\ X_c = DX_b + DX_d; \\ X_d = DX_b + DX_d; \\ X_e = D^2 X_c. \end{cases} \tag{3.357}$$

Записана система лінійних рівнянь (3.357) є повною та несуперечливою. Може скластися враження, що ця система є виродженою, оскільки у третьому та четвертому рівняннях для значень X_c та X_d записані еквівалентні аналітичні вирази, але тут необхідно звернути увагу на те, що ці рівняння не є взаємозалежними.

Тепер, на основі отриманої системи рівнянь (3.357), необхідно знайти передавальну функцію декодера, яка у загальному випадку записується через співвідношення вихідного сигналу до вхідного, тобто [33, 52, 53, 55, 62, 63]:

$$T(D) = \frac{X_e}{X_a}. \tag{3.358}$$

Розв’яжемо систему рівнянь (3.357) наступним чином. Спочатку перетворимо перше рівняння цієї системи, виразивши змінну X_a через змінні X_b та X_c. Відповідне співвідношення буде мати наступний вигляд:

$$X_a = \frac{X_b - X_c}{D^2}. \tag{3.359}$$

Враховуючи отримане співвідношення (3.359) та четверте рівняння

системи (3.357), можна записати аналітичний вираз для передавальної функції $T(D)$, заданої співвідношенням (3.358), наступним чином:

$$T(D) = \frac{X_e}{X_a} = \frac{D^2 X_c}{\frac{X_b - X_c}{D^2}} = \frac{D^2 D^2 X_c}{X_b - X_c} = \frac{D^4 X_c}{X_b - X_c}. \quad (3.360)$$

Тепер перепишемо друге та третє рівняння системи (3.357) таким чином, щоб надалі знайти аналітичні вирази для змінних та X_b та X_c через змінну X_d. Відповідні аналітичні вирази можна записати у вигляді наступної системи лінійних рівнянь:

$$\begin{cases} \frac{X_d}{D} = X_b + X_d, \\ \frac{X_c}{D} = X_b + X_d. \end{cases} \quad (3.361)$$

З першого рівняння системи (3.361) отримуємо аналітичний вираз, в якому змінна X_b явно виражена через змінну X_d та через параметр просвіту D:

$$X_b = \frac{X_d}{D} - X_d = X_d \left(\frac{1}{D} - 1\right). \quad (3.362)$$

Аналогічно, перепишемо третє рівняння системи (3.361) та, враховуючи отримане співвідношення (3.362), отримаємо аналітичний вираз, який характеризує залежність $X_c(X_d)$. Відповідно, маємо:

$$X_c = D\left(X_d\left(\frac{1}{D} - 1\right)\right) + DX_d = X_d\left(D\left(\left(\frac{1}{D} - 1\right)\right) + D\right) =$$

$$= X_d D\left(\left(\frac{1}{D} - 1\right) + 1\right) = X_d,$$

тобто, остаточно:

$$X_c = X_d. \quad (3.363)$$

Отримана в результаті аналітичних перетворень тотожність (3.363) безпосередньо випливає з того, що аналітичні вирази для змінних X_c та X_d у початковій системі рівнянь (3.357) є однаковими.

Тепер, з урахуванням отриманих співвідношень (3.362) та (3.363), можна переписати аналітичний вираз (3.360) для передавальної функції $T(D)$ через параметр просвіту D та змінну X_d наступним чином:

$$T(D) = \frac{X_e}{X_a} = \frac{D^4 X_c}{X_b - X_c} = \frac{D^4 X_d}{X_d\left(\frac{1}{D} - 1\right) - X_d} = \frac{D^4 X_d}{X_d\left(\left(\frac{1}{D} - 1\right) - 1\right)} =$$

$$= \frac{D^4}{\frac{1}{D} - 2} = \frac{D^4}{\frac{1}{D} - \frac{2D}{D}} = \frac{D^4}{\frac{1 - 2D}{D}} = \frac{D^5}{1 - 2D}.$$

Тобто, остаточно передавальна функція $T(D)$ декодера згорткового коду з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$ залежить лише від параметру просвіту D та відповідний аналітичний вираз має наступний вигляд:

$$T(D) = \frac{D^5}{1-2D}. \qquad (3.364)$$

У співвідношенні (3.364) аналітичний вираз для функції $T(D)$ записаний у вигляді раціонального дробу. Перепишемо тепер це співвідношення у поліноміальній формі з використанням теорії твірних функцій, яка була описана у підрозділі 3.8 першого тому другої частини посібника [48]. Враховуючи те, що степеневий ряд Тейлора для дрібно-раціональної функції $f(x) = \frac{1}{1-ax}$ має вигляд [48]:

$$f(x) = \frac{1}{1-ax} = 1 + ax + a^2x^2 + a^3x^3 + \cdots + a^nx^n + \cdots \qquad (3.365)$$

співвідношення (3.364), з урахуванням (3.365), можна переписати у поліноміальній формі наступним чином:

$$T(D) = D^5 + 2D^6 + 4D^7 + \cdots + 2^l D^{l+5} + \cdots, \quad l \in N. \qquad (3.366)$$

Фізичний зміст отриманого співвідношення (3.366) полягає у тому, що воно повністю описує ґраткову діаграму згорткового коду з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$, наведену на рис. 3.103. Згідно із співвідношенням (3.366) існує лише 1 шлях із метрикою 5, 2 – із метрикою 6, 4 – із метрикою 7. Загалом для будь-якого натурального числа l існує 2^l шляхів, метрика яких складає $l + 5$. Тобто, в даному підрозділі на основі теорії скінченних автоматів та теорії твірних функцій строго математично обґрунтувано, що для згорткового коду з

параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$, структурна схема цифрового кодувального пристрою для якого представлена на рис. 3.63, параметр просвіту d_f становить 5.

Проте узагальнена схем скінченного автомату, яка наведена на рис. 3.105, а також передавальна функція системи $T(D)$, яка описує цей скінченний автомат та визначається співвідношеннями (3.364), (3.366), не дають повної інформації про можливі помилки у згортковому коді. Для проведення більш точних оцінок з використанням методів дискретної математики [48 – 50] вводяться додаткові параметри лічильників гілок [33, 52, 53, 55, 62, 63], які розглядатимуться у наступному підрозділі.

3.8.13.3 Лічильники гілок та узагальнена форма передавальної функції декодеру згорткового коду

З використанням передавальної функції системи декодування послідовностей згорткового коду, побудованої з використанням теорії скінченних автоматів, можна отримати значно більше корисної інформації, ніж у разі формування метрик гілок діаграми станів скінченного автомату лише через відстань D між нульовою та очікуваною кодовими комбінаціями. Введемо для гілок діаграми станів два додаткових параметри, які у літературі з теорії згорткових кодів називаються лічильниками гілок [33, 52, 53, 55, 62, 63]. До кожної гілки діаграми станів скінченного автомату додаймо множник L таким чином, щоб цей параметр був лічильником гілок між початковим та кінцевим станами автомату $a = 00$ та $e = 00$. Крім цього, введемо додатковий параметр N для гілок всіх переходів між станами скінченного автомату, які породжені двійковим вхідним сигналом 1. Згідно з умовними позначеннями, введеними на рис. 3.64, такі гілки на діаграмі станів відображені пунктирними лініями. У разі введення параметрів лічильників гілок схема скінченного автомату, наведена на рис. 3.105, буде мати інший вигляд. Така схема із відповідними змінами показана на рис. 3.106.

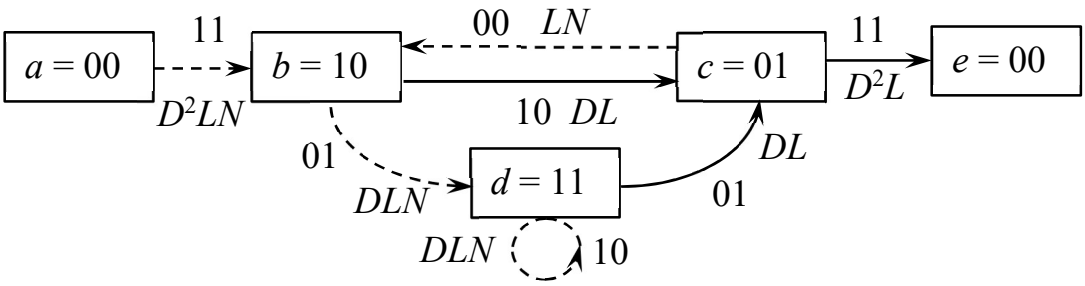


Рис. 3.106 Схема скінченного автомату для кодеру згорткового коду з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$ із вказаними метриками гілок та визначеними параметрами лічильників гілок

З урахуванням введений параметрів лічильників гілок систему лінійних рівнянь (3.357), яка визначає передавальну функцію декодеру згорткового коду, можна переписати у наступному вигляді [33]:

$$\begin{cases} X_b = D^2LN\ _a + LNX_c; \\ X_c = DLX_b + DLX_d; \\ X_d = DLNX_b + DLNX_d; \\ X_e = D^2LX_c. \end{cases} \tag{3.367}$$

Система рівнянь (3.367) розв'язується аналогічно системі (3.357). Спочатку, з першого рівняння системи (3.367), запишемо аналітичний вираз для змінної X_a:

$$X_a = \frac{X_b - LNX_c}{LN^{\ 2}}. \tag{3.368}$$

За такої умови передавальна функція кодувальної системи записується через змінні X_b та X_c наступним чином:

$$T(D, L, N) = \frac{X_e}{X_a} = \frac{D^2LX_c}{\frac{X_b - LNX_c}{LND^2}} = \frac{L^2D^4NX_c}{X_b - LNX_c}. \tag{3.369}$$

Далі, з третього рівняння системи (3.367) отримуємо аналітичний вираз, в якому змінна X_b явно виражена через змінну X_d та через параметри *D, L* та *N*:

$$X_b = \frac{X_d}{DLN} - X_d = X_d\left(\frac{1}{DLN} - 1\right). \tag{3.370}$$

Аналогічно, перепишемо друге рівняння системи (3.367) та, враховуючи отримане співвідношення (3.370), отримаємо аналітичний вираз, який

характеризує залежність $X_c(X_d)$. Відповідно, маємо:

$$X_c = DL\left(X_d\left(\frac{1}{DLN}-1\right)\right) + DL_{\ d} = X_d\left(DL\left(\left(\frac{1}{DLN}-1\right)\right)+DL\right) =$$

$$= X_d DL\left(\left(\frac{1}{DLN}-1\right)+1\right) = \frac{X_d}{N},$$

тобто, остаточно:

$$X_c = \frac{X_d}{N}. \tag{3.371}$$

Підставляючи отримані співвідношення (3.370) та (3.371) в аналітичний вираз для передавальної функції T(D, L, N), заданий співвідношенням (3.369), отримуємо наступний аналітичний вираз:

$$T(D,L,N) = \frac{L^2D^4NX_c}{X_b - LNX_c} = \frac{L^2D^4N\frac{X_d}{N}}{X_d\left(\frac{1}{DLN}-1\right) - LN\frac{X_d}{N}} = \frac{L^2D^4X_d}{X_d\left(\frac{1}{DLN}-1\right) - LX_d} =$$

$$= \frac{L^2D^4}{\left(\frac{1}{DLN}-1\right)-L} = \frac{L^2D^4}{\left(\frac{1-DLN}{DLN}\right)-L} = \frac{L^2D^4}{\left(\frac{1-DLN}{DLN}\right)-\frac{DL^2N}{DLN}} =$$

$$= \frac{L^2D^4}{\frac{1-DLN-DL^2N}{DLN}} = \frac{D^5L^3N}{1-DLN-DL^2N} = \frac{D^5L^3N}{1-DLN(1+L)}.$$

Тобто, маємо остаточний вираз для передавальної функції декодеру згорткового коду з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$:

$$T(D,L,N) = \frac{D^5L^3N}{1-DLN(1+L)}. \tag{3.372}$$

Враховуючи степеневий ряд Тейлора (3.365) для дрібно-раціональної функції $f(x) = \frac{1}{1-ax}$, співвідношення (3.372) можна переписати у поліноміальній формі наступним чином [33]:

$$T(D,L,N) = \frac{D^5L^3N}{1-DLN(1+L)} = D^5L^3N + D^6L^4(1+L)N^2 +$$

$$+D^7L^5(1+L^2)N^3 + \ldots + D^{l+5}L^{l+3}N^{l+1} + \cdots. \tag{3.373}$$

Аналіз отриманого співвідношення (3.373) дає наступні результати. Зрозуміло, що існує лише 1 шлях з метрикою 5, якому відповідає множник D^5.

Згідно з множником L^3 довжина цього шляху складає 3 гілки, та згідно із множником N лише одна з цих трьох гілок відповідає вхідному сигналу m =1. Проаналізуємо другий доданок співвідношення (3.373). Цей аналіз показує, що існує 2 шляхи із кодовою відстанню 6. Довжина одного з цих шляхів складає 4 гілки, а другого – 5 гілок, але обидві ці шляхи відрізняються від нульового лише двома вхідними бітами. Також існують чотири шляхи із кодовою відстанню 7, довжина одного з них складає 5 гілок, два мають довжину 6 гілок, а довжина останнього, четвертого шляху, становить 7 гілок. Проте загальний множник N^3 свідчить про те, що усім чотирьом шляхам із кодовою відстанню 7 відповідають вхідні послідовності із трьома одиничними бітами.

Розглянутий у цьому підрозділі приклад показує, що введення параметрів лічильників гілок до передавальної функції декодеру згорткового коду дозволяє проводити аналіз можливих помилок, які виникають на хибних шляхах роботи декодеру. Наприклад, на основі проведеного аналізу зрозуміло, що якщо кодова відстань між нульовою та прийнятою комбінаціями згорткового коду становить 7, така послідовність містить 3 помилкових кодових слова, яким відповідають значення вхідних бітів m =1.

Тобто, аналіз алгоритму роботи скінченного автомату, що описує декодер згорткового коду та кодувально-декодуваьлну систему, дозволяє через аналіз параметрів передавальної функції оцінити $T(D, L, N)$ оцінити її коерктувальну здатність. Приклади такого аналізу для більш складних декодерів, з іншими параметрами, будуть наведені у наступному підрозділі.

3.8.13.4 Приклади отримання аналітичних виразів для передавальних функцій згорткових кодів з іншими параметрами

На основі аналізу прикладів формування передавальних функцій декодеру згорткового коду з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$ через аналіз схеми скінченного автомату, сформулюємо головні правила формування таких функцій.

1. Проводиться перетворення схеми скінченного автомату згідно з

загальними правилами, сформульованими у підрозділі 3.8.13.2.

2. Через вагу вихідних кодових комбінацій, які відповідають гілкам переходів між станами автомату, визначаються степені коефіцієнта *D*.

3. До ваги кожної гілки додається множник *L*, який називається лічильником гілок. Таким чином, число перед множником *L* у аналітичному виразі для передавальної функції відповідає кількості гілок у шляху заданої ваги.

3. Якщо перехід між станами здійснюється за умови дії вхідного сигналу 1, до ваги такої гілки додається множник *N*, який визначає кількість гілок у обраному шляху, що відповідають хибному вхідному сигналу.

Загалом описаний алгоритм є простим. Проблема полягає у тому, що у разі складних схем скінченного автомату аналітичні вирази для передавальної функції становляться вкрай громіздкими і аналізувати їх ручним способом стає вкрай складним. Така проблема виникає навіть для декодеру з параметрами $K = 4$, $\frac{1}{n} = \frac{1}{3}$, розглянутого у підрозділі 3.8.12.2. Узагальнена структурна схема цього кодеру наведена на рис. 3.96. Саме з цієї причини аналіз коректувальних здатностей згорткових кодів сьогодні зазвичай здійснюється з використанням сучасних засобів обчислювальної техніки [33, 62, 63]. Наприклад, для пошуку розкладання передавальної функції в степеневий ряд з використанням математичного апарату твірних функцій можуть бути використані сучасні комп'ютерні системи для проведення аналітичних перетворень та символьні процесори. Відповідні функції символьного процесору системи науково-технічних розрахунків MatLab та приклади їхнього використання розглядалися у першій частині другого тому цього посібника [48].

Розглянемо деякі приклади пошуку передавальної функції декодеру згорткового коду.

Приклад 3.69 Через аналіз схеми скінченного автомату, наведеної на

рис. 3.97, знайти аналітичний вираз для передавальної функції декодеру згорткового коду з параметрами $K = 4$, $\frac{1}{n} = \frac{1}{3}$, узагальнена структурна схема якого наведена на рис. 3.96, та визначити мінімальну кодову відстань цього коду.

Спочатку проведемо перетворення схеми автомату, наведеної на рис. 3.96, згідно з правилами, сформульованими у підрозділі 3.8.12.2. Тобто, видалимо з вузла *a* петльову гілку та введемо додатковий вихідний вузол *i* з нульовим станом, до якого йде одна гілка з вузла *e*. Відповідна перетворена схема скінченного автомату, на якій також вказані метрики гілок, наведена на рис. 3.107.

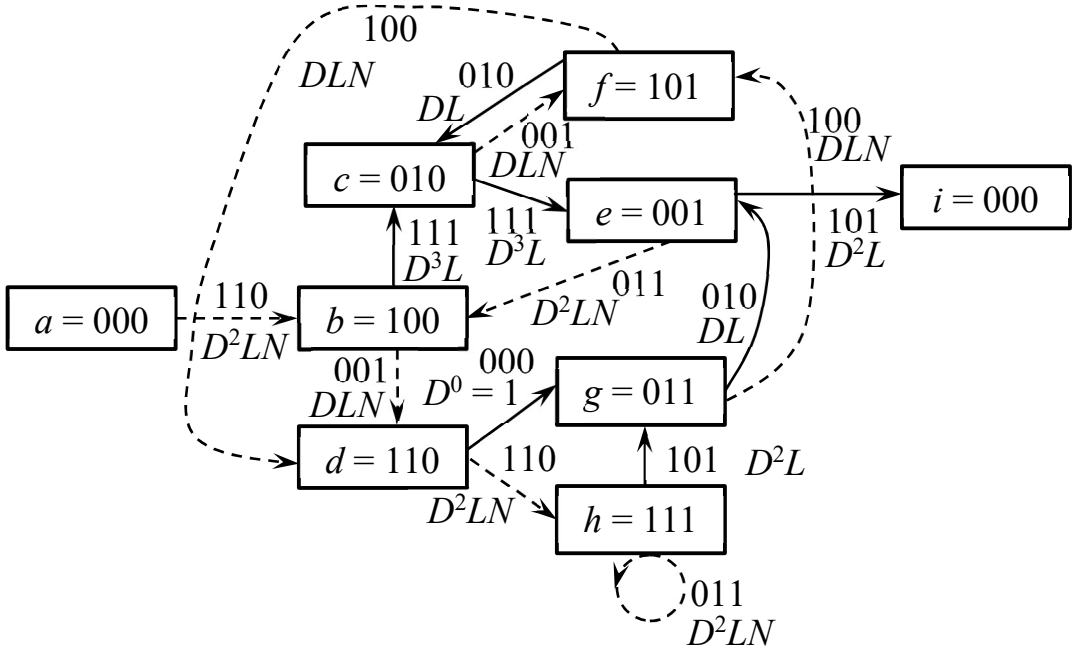


Рис. 3.107 Перетворена схема скінченного автомату для декодеру згорткового коду з параметрами $K = 4$ та $\frac{1}{n} = \frac{1}{3}$ із вказаними метриками гілок для прикладу 3.69

7

Для отримання передавальної функції декодеру згорткового коду з параметрами $K = 4$ та $\frac{1}{n} = \frac{1}{3}$ з згідно із діаграмою станів скінченного автомату, наведеною на рис. 3.107, можна записати наступну систему з восьми лінійних рівнянь:

$$\begin{cases} X_b = D^2LN(X_a + X_e); \\ X_c = D^3LX_b + DL_f; \\ X_d = DLN(X_b + X_f); \\ X_e = D^3LX_c + DLX_g; \\ X_f = DLN(X_g + X_c); \\ X_g = X_d + D^2LX_h; \\ X_h = D^2LN(X_d + X_h); \\ X_i = D^2LX_e. \end{cases} \quad (3.374)$$

Послідовно розв'яжемо систему рівнянь (3.374) та знайдемо аналітичний вираз для передавальної функції системи $\frac{X_i}{X_a}$. З першого рівняння цієї системи запишемо наступний вираз для змінної X_a:

$$X_a = \frac{X_b}{D^2LN} - X_e. \quad (3.375)$$

З останнього, восьмого рівняння системи (3.374), з урахуванням виразу (3.375), можна записати:

$$\frac{X_i}{X_a} = \frac{D^2LX_e}{\frac{X_b}{D^2LN} - X_e}. \quad (3.376)$$

Розв'язуючи сьоме рівняння системи (3.374) отримуємо наступний аналітичний вираз для змінної X_d через змінну X_h:

$$X_d = X_h\left(\frac{1}{D^2LN} - 1\right). \quad (3.377)$$

З урахуванням отриманого співвідношення (3.377) можна, підставляючи отримане значення X_d до шостого рівняння системи (3.374), отримати аналітичний вираз для змінної X_g через змінну X_h:

$$X_g = X_d + D^2LX_h = X_h\left(\frac{1}{D^2LN} - 1\right) + D^2LX_h =$$

$$= X_h\left(\frac{1}{D^2LN} + D^2L - 1\right). \quad (3.378)$$

Головною метою аналітичних перетворень, які проводяться, є формування математичного виразу зі зменшеною кількістю проміжних змінних, від яких залежать всі стани автомату. Остаточно необхідно знайти аналітичний вираз для змінних X_b та X_e через змінну X_h. За такої умови,

підставляючи знайдені аналітичні функції у співвідношення для передавальної функції (3.376), можна отримати шуканий аналітичний вираз $T(D, L, N)$. З урахуванням поставленої мети щодо проведення аналітичних перетворень та отриманих співвідношень (3.377), (3.378), перепишемо четверте та п'яте рівняння системи (3.374) наступним чином:

$$X_f = DLN(X_g + X_c) = DLN\left(X_h\left(\frac{1}{D^2LN} + D^2L - 1\right) + X_c\right). \quad (3.379)$$

$$X_e = D^3LX_c + DLX_g = D^3LX_c + DL\left(\frac{1}{D^2LN} + D^2L - 1\right)X_h. \quad (3.380)$$

Перевага отриманих співвідношень (3.379), (3.380) полягає у тому, що стани скінченного автомату X_f та X_e тепер залежать лише від двох станів: X_c та X_h, проте змінну X_c в остаточно сформованому виразі для стану X_e також треба виключити. Тобто, остаточні вирази повинні бути записані у формі $X_b = X_b(X_h)$ та $X_e = X_e(X_h)$.

Тепер, скориставшись отриманими співвідношеннями (3.377) та (3.379) для станів X_d та X_f, перепишемо третє рівняння системи (3.374) наступним чином:

$$X_h\left(\frac{1}{D^2LN} - 1\right) = DLN(X_b + X_f) =$$

$$= DLN\left(X_b + DLN\left(X_h\left(\frac{1}{D^2LN} + D^2L - 1\right) + X_c\right)\right),$$

або, після відповідних аналітичних перетворень:

$$X_h\left(\frac{1}{D^2LN} - 1\right) = DLNX_b + DLNX_c + D^2L^2N^2\left(\frac{1}{D^2LN} + D^2L - 1\right)X_h,$$

тобто,

$$DLN(X_b + X_c) = \left(\frac{1}{D^2LN} - 1 - \frac{D^2L^2N^2}{D^2LN} - D^4L^3N^2 - D^2L^2N^2\right)X_h =$$

$$= \left(\frac{1}{D^2LN} - 1 - LN - D^4L^3N^2 - D^2L^2N^2\right)X_h. \quad (3.381)$$

Отримане співвідношення (3.381) є аналітичним виразом для суми станів декодеру $X_b + X_c$ залежно від стану X_h та параметрів D, L та N. Продовжимо розв'язування поставленого раніше завдання, тобто, будемо шукати пряму

аналітичну залежність $X_b(X_h)$. За умови наявності такої залежності можна буде скористатися співвідношеннями (3.380), (3.381) для формування прямої аналітичної залежності $X_e(X_h)$, тобто, завдання формування аналітичних виразів для передавальної функції згорткового коду (3.376) буде остаточно розв'язаним.

Для пошуку аналітичної залежності $X_b(X_h)$ скористаємося другим рівнянням системи (3.374), підставляючи в нього замість змінної X_f вираз (3.379). Відповідний аналітичний вираз можна записати наступним чином:

$$X_c = D^3LX_b + DL\ _f = D^3LX_b + DL\left(DLN\left(X_h\left(\frac{1}{D^2LN} + D^2L - 1\right) + X_c\right)\right),$$

або, після відповідних аналітичних перетворень:

$$X_c = D^3LX_b + DL\ _f = D^3LX_b + D^2L^2N\left(X_h\left(\frac{1}{D^2LN} + D^2L - 1\right) + X_c\right) =$$

$$= D^3LX_b + D^2L^2NX_c + D^2L^2N\left(\frac{1}{D^2LN} + D^2L - 1\right)X_h =$$

$$= D^3LX_b + D^2L^2NX_c + D^2L^2N\left(\frac{1}{D^2LN} + D^2L - 1\right)X_h =$$

$$= D^3LX_b + D^2L^2NX_c + (L + D^4L^3N - D^2L^2N)X_h. \qquad (3.382)$$

Розв'язуючи отримане рівняння (3.382) відносно змінної стану автомату X_c, отримуємо аналітичну функцію $X_c(X_b, X_h)$, записану в явній формі:

$$X_c = \frac{D^3LX_b + (L + D^4L^3N - D^2L^2N)X_h}{1 - D^2L^2N}. \qquad (3.383)$$

Зрозуміло, що система рівнянь (3.381), (3.383) є зімкненою відносно змінних X_b, X_c, та X_h. Тоді, підставляючи отриманий аналітичний вираз (3.383) до співвідношення (3.381), можна отримати шукану залежність для станів скінченного автомату $X_b(X_h)$ у явній формі. Відповідні аналітичні перетворення мають наступний вигляд:

$$DLN\left(X_b + \frac{D^3LX_b + (L + D^4L^3N - D^2L^2N)X_h}{1 - D^2L^2N}\right) =$$

$$= \left(\frac{1}{D^2LN} - 1 - LN - D^4L^3N^2 - D^2L^2N^2\right)X_h. \qquad (3.384)$$

Розкриваючи дужки в лівій частині рівняння (3.384), розділимо змінні X_b та X_h. Відповідно, маємо:

$$DLNX_b\left(1+\frac{D^3L}{1-D^2L^2N}\right)+DLN\frac{L+D^4L^3N-D^2L^2N}{1-D^2L^2N}\,X_h=$$

$$=\left(\frac{1}{D^2LN}-1-LN-D^4L^3N^2-D^2L^2N^2\right)X_h,$$

або, після ділення обох частин отриманого рівняння на постійний множник DLN,

$$X_b\left(1+\frac{D^3L}{1-D^2L^2N}\right)+\frac{L+D^4L^3N-D^2L^2N}{1-D^2L^2N}\,X_h=$$

$$=\frac{\frac{1}{D^2LN}-1-LN-D^4L^3N^2-D^2L^2N^2}{DLN}X_h. \qquad (3.385)$$

Явний аналітичний вираз $X_b(X_h)$ після перенесення всіх доданків, які стоять перед змінною X_h, до правої частини рівняння (3.385), зведення подібних доданків та зведення до спільного знаменника, записується наступним чином:

$$X_b\left(1+\frac{D^3L}{1-D^2L^2N}\right)=$$

$$=\left(\frac{1}{D^2LN}-1-LN-D^4L^3N^2-D^2L^2N^2-\frac{L+D^4L^3N-D^2L^2N}{1-D^2L^2N}\right)X_h=$$

$$=\left(\frac{1-D^2LN-D^2L^2N^2-D^6L^4N^3-D^4L^3N^3}{D^3L^2N^2}-\frac{L+D^4L^3N-D^2L^2N}{1-D^2L^2N}\right)X_h=$$

$$=\left(\frac{1-D^2LN(1+LN)-D^4L^3N^3-D^6L^4N^3}{D^3L^2N^2}-\frac{L+D^4L^3N-D^2L^2N}{1-D^2L^2N}\right)X_h=$$

$$=\left(\frac{1-D^2LN(1+LN)-D^4L^3N^3-D^6L^4N^3-D^2L^2N}{D^3L^2N^2(1-D^2L^2N)}+\right.$$

$$\left.+\frac{D^4L^3N^2(1+LN)+D^6L^5N^4+D^8L^6N^4}{D^3L^2N^2(1-D^2L^2N)}-\frac{L+D^4L^3N-D^2L^2N}{1-D^2L^2N}\right)X_h=$$

$$=\left(\frac{1-D^2LN(1+LN)-D^4L^3N^3-D^6L^4N^3-D^2L^2N}{D^3L^2N^2(1-D^2L^2N)}+\right.$$

$$+\frac{D^4L^3N^2(1+LN)+D^6L^5N^4+D^8L^6N^4}{D^3L^2N^2(1-D^2L^2N)}-$$

$$-\frac{D^3L^3N^2-D^7L^5N^3+D^5L^4N^3}{D^3L^2N^2(1-D^2L^2N)}\Bigg)X_h=$$

$$=\left(\frac{1-D^2LN\big(1+L(N+1)\big)-D^4L^3N^2\big(1+N(L+1)\big)-D^6L^4N^3(1+LN)}{D^3L^2N^2(1-D^2L^2N)}+\right.$$

$$+\frac{D^8L^6N^4-D^3L^3N^2+D^7L^5N^3-D^5L^4N^3}{D^3L^2N^2(1-D^2L^2N)}\Bigg)X_h=$$

$$=\left(\frac{D^8L^6N^4+D^7L^5N^3+D^6L^4N^3(1+LN)-D^5L^4N^3}{D^3L^2N^2(1-D^2L^2N)}-\right.$$

$$-\frac{D^4L^3N^2\big(1+N(L+1)\big)+D^3L^3N^2+D^2LN\big(1+L(N+1)\big)-1}{D^3L^2N^2(1-D^2L^2N)}\Bigg)X_h. \qquad (3.386)$$

З урахуванням отриманого дрібно-раціонального співвідношення (3.386), остаточний вираз для аналітичної функції $X_b(X_h)$ можна записати наступним чином:

$$X_b(X_h)=\left(\frac{D^8L^6N^4+D^7L^5N^3+D^6L^4N^3(1+LN)-D^5L^4N^3}{D^3L^2N^2(1-D^2L^2N)}-\right.$$

$$-\frac{D^4L^3N^2\big(1+N(L+1)\big)+D^3L^3N^2+D^2LN\big(1+L(N+1)\big)-1}{D^3L^2N^2(1-D^2L^2N)}\Bigg)x$$

$$x\left(\frac{1-D^2L^2N}{1-D^2L^2N+D^3L}\right)X_h=\frac{D^8L^6N^4+D^7L^5N^3+D^6L^4N^3(1+LN)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h-$$

$$-\frac{D^5L^4N^3+D^4L^3N^2\big(1+N(L+1)\big)+D^3L^3N^2+D^2LN\big(1+L(N+1)\big)-1}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h-$$

$$-\frac{D^{10}L^8N^4+D^9L^7N^4+D^8L^6N^4(1+LN)-D^7L^6N^4}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h+$$

$$+\frac{D^6L^5N^3\big(1+N(L+1)\big)+D^5L^5N^3+D^4L^3N^2\big(1+L(N+1)\big)-D^2L^2N}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h=$$

$$=\frac{-D^{10}L^8N^4+D^9L^7N^4+D^8L^6N^4(LN+2)+D^7L^5N^3(1-LN)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h+$$

$$+\frac{D^6L^4N^3(1+(2N+1)L+L^2N)+D^5L^4N^3(L-1)+D^4L^3N^2(L-N)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h-$$

$$-\frac{D^3L^3N^2-D^2LN(1+LN)+1}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h. \quad (3.387)$$

Маючи явний аналітичний вираз для функції $X_b(X_h)$, записаний у вигляді співвідношення (3.387), можна, виключаючи із співвідношення (3.383) змінну стану скінченного автомату X_b, отримати аналітичний вираз для залежності $X_c(X_h)$. Відповідні аналітичні перетворення мають наступний вигляд:

$$X_c = \frac{D^3LX_b+ (L+D^4L^3N-D^2L^2N)X_h}{1-D^2L^2N} = \frac{X_{cn}}{X_{cd}};$$

$$X_{cn}(X_h) = D^3LX_b + (L + D^4L^3N - D^2L^2N)X_h =$$

$$= D^3L\left(\frac{-D^{10}L^8N^4 + D^9L^7N^4 + D^8L^6N^4(LN+2)+D^7L^5N^3(1-LN)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)} + \right.$$

$$+\frac{D^6L^4N^3(1+(2N+1)L+L^2N)+D^5L^4N^3(L-1)+D^4L^3N^2(L-N)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)} -$$

$$\left. -\frac{D^3L^3N^2-D^2LN(1+LN)+1}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}\right)X_h + (L + D^4L^3N - D^2L^2N)X_h =$$

$$= \frac{-D^{13}L^9N^4 + D^{12}L^8N^4 + D^{11}L^7N^4(LN+2)+D^{10}L^6N^3(1-LN)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h +$$

$$+\frac{D^9L^5N^3(1+(2N+1)L+L^2N)+D^8L^5N^3(L-1)+D^7L^4N^2(L-N)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h -$$

$$-\frac{D^6L^4N^2-D^5L^2N(1+LN)+1}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h + (L + D^4L^3N - D^2L^2N)X_h. \quad (3.388)$$

Для запису отриманого аналітичного виразу $X_{cn}(X_h)$ у вигляді раціонального дробу введемо ще одну додаткову змінну:

$$X_{ch} = L + D^4L^3N - D^2L^2N =$$

$$= \frac{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)(L+D^4L^3N-D^2L^2N)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)} =$$

$$= \frac{(1-D^5L^3N^3)(1-D^2L^2N+D^3L)(L+D^4L^3N-D^2L^2N)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)} =$$

$$= \frac{(1-D^2L^2N+D^3L-D^5L^3N^3+D^7L^5N^4-D^8L^4N^3)(L-D^2L^2N+D^4L^3N)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)} =$$

$$= \frac{L-D^2L^3N+D^3L^2-D^5L^4N^3+D^7L^6N^4-D^8L^5N^3-D^2L^2N+D^4L^4N^2-D^5L^3N}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)} +$$

$$+\frac{D^7L^5N^4-D^9L^7N^5+D^{10}L^6N^4+D^4L^3N-D^6L^5N^2+D^7L^4N-D^9L^6N^4}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}+$$

$$+\frac{D^{11}L^8N^5-D^{12}L^7N^4}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}=\frac{-D^{12}L^7N^4+D^{11}L^8N^5+D^{10}L^6N^4}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}-$$

$$-\frac{D^9L^6N^4(LN+1)+D^8L^5N^3-D^7L^4N(L^2N^3+LN^3+1)+D^6L^5N^2+D^5L^3N(LN^2+1)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}+$$

$$+\frac{D^4L^3N(LN+1)+D^3L^2-D^2L^2N(L+1)+L}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}. \quad (3.389)$$

Враховуючи отримані дрібно-раціональні вирази (3.388) та (3.389) та зводячи подібні доданки перед рівними степенями змінної D, отримуємо остаточну аналітичну залежність $X_c(X_h)$ у наступному вигляді:

$$X_c(X_h)=D^3LX_b+X_{ch}X_h=\frac{-D^{13}L^9N^4+D^{12}L^7N^4(L-1)+2D^{11}L^7N^4(LN+1)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}+$$

$$+\frac{D^{10}L^6N^3\big(1-N(L+1)\big)+D^9L^5N^3\big(1+(N+1)L-L^2N(N-1)\big)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}+$$

$$+\frac{D^8L^5N^3(L-2)+D^7L^4N\big(L(L+1)N^3-N^2+LN+1\big)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}+$$

$$+\frac{D^6L^4N^2(1-2L)-D^5L^2N(1+L(N+1)+L^2N^2)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}+$$

$$+\frac{D^4L^3N(LN+1)+D^3L^2-D^2L^2N(L+1)+L+1}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h. \quad (3.390)$$

Маючи залежність $X_c(X_h)$, записану у вигляді співвідношення (3.390), з використанням співвідношення (3.380) можна знайти аналітичну залежність $X_e(X_h)$. Відповідні співвідношення можна записати у наступному вигляді:

$$X_e=D^3LX_c+DL\left(\frac{1}{D^2LN}+D^2L-1\right)X_h=$$

$$=D^3LX_c+\frac{D^2L^2N+D^6L^4N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h+$$

$$+\frac{D^4L^3N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h. \quad (3.391)$$

Аналізуючи співвідношення (3.391), розглянемо окремо знаменник цього дробу та розкриємо дужки. Відповідно маємо:

$$(1-D^2L^2N)(1-D^2L^2N+D^3L)=1-D^2L^2N+D^3L-D^2L^2N+D^4L^4N^2+D^5L^3N=$$

$$= 1 - 2D^2L^2N + D^3L + D^4L^4N^2 + D^5L^3N. \quad (3.392)$$

З урахуванням (3.392), перепишемо аналітичний вираз $X_e(X_h)$, заданий співвідношенням (3.391), наступним чином:

$$X_e(X_h) = D^3LX_c + \frac{D^2L^2N + D^6L^4N^2(1 - 2D^2L^2N + D^3L + D^4L^4N^2 + D^5L^3N)}{D^3L^2N^2(1 - 2D^2L^2N + D^3L + D^4L^4N^2 + D^5L^3N)}X_h =$$

$$= \frac{-D^{16}L^{10}N^4 + D^{15}L^8N^4(L-1) + 2D^{14}L^8N^4(LN+1)}{D^3L^2N^2(1 - D^2L^2N)(1 - D^2L^2N + D^3L)}X_h +$$

$$+ \frac{D^{13}L^7N^3\big(1 - N(L+1)\big) + D^{12}L^6N^3\big(1 + (N+1)L - L^2N(N-1)\big)}{D^3L^2N^2(1 - D^2L^2N)(1 - D^2L^2N + D^3L)}X_h +$$

$$+ \frac{D^{11}L^6N^3(L-2) + D^{10}L^5N\big(L(L+1)N^3 - N^2 + LN + 1\big)}{D^3L^2N^2(1 - D^2L^2N)(1 - D^2L^2N + D^3L)}X_h +$$

$$+ \frac{D^9L^5N^2(1-2L) - D^8L^3N(1 + L(N+1) + L^2N^2)}{D^3L^2N^2(1 - D^2L^2N)(1 - D^2L^2N + D^3L)}X_h +$$

$$+ \frac{D^7L^4N(LN+1) + D^6L^3 - D^5L^3N(L+1) + D^3L^2 + D^3L}{D^3L^2N^2(1 - D^2L^2N)(1 - D^2L^2N + D^3L)}X_h +$$

$$+ \frac{D^2L^2N + D^6L^4N^2 - 2D^8L^6N^3 + D^9L^5N^2 + D^{10}L^8N^4 + D^{11}L^7N^3}{D^3L^2N^2(1 - 2D^2L^2N + D^3L + D^4L^4N^2 + D^5L^3N)}X_h =$$

$$= \frac{-D^{16}L^{10}N^4 + D^{15}L^8N^4(L-1) + 2D^{14}L^8N^4(LN+1)}{D^3L^2N^2(1 - D^2L^2N)(1 - D^2L^2N + D^3L)}X_h +$$

$$+ \frac{D^{13}L^7N^3\big(1 - N(L+1)\big) + D^{12}L^6N^3\big(1 + (N+1)L - L^2N(N-1)\big)}{D^3L^2N^2(1 - D^2L^2N)(1 - D^2L^2N + D^3L)}X_h +$$

$$+ \frac{2D^{11}L^6N^3(L-1) + D^{10}L^5N\big(L^3N^3 + L(L+1)N^3 - N^2 + LN + 1\big)}{D^3L^2N^2(1 - D^2L^2N)(1 - D^2L^2N + D^3L)}X_h +$$

$$+ \frac{2D^9L^5N^2(1-L) + D^8L^3N(2L^3N^2 - L^2N^2 - L(N+1) - 1)}{D^3L^2N^2(1 - D^2L^2N)(1 - D^2L^2N + D^3L)}X_h +$$

$$+ \frac{D^7L^4N(LN+1) + D^6L^3(LN^2+1) - D^5L^3N(L+1) + D^3L^2 + D^3L + D^2L^2N}{D^3L^2N^2(1 - D^2L^2N)(1 - D^2L^2N + D^3L)}X_h. \quad (3.393)$$

Тепер, аналізуючи знаменник дробу (3.376), запишемо аналітичний вираз для знаменника дробу, який описує передавальну функцію згорткового коду $T(D)$. Відповідно, маємо:

$$T_d(X_b, X_e) = \frac{X_b}{D^2LN} - X_e = \frac{X_b - D^2LNX_e}{D^2LN}. \quad (3.394)$$

Враховуючи отримані співвідношення (3.387) та (3.393) для станів системи X_b та X_e, виражених через стан X_h, перепишемо співвідношення (3.394) таким чином, щоб знайти залежність $T_d(X_h)$ у явному вигляді. Відповідно, маємо:

$$T_d(X_h) = \frac{X_b - D^2LNX_e}{D^2LN} =$$

$$= \frac{-D^{10}L^8N^4 + D^9L^7N^4 + D^8L^6N^4(LN+2) + D^7L^5N^3(1-LN)}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h +$$

$$+ \frac{D^6L^4N^3(1+(2N+1)L+L^2N) + D^5L^4N^3(L-1) + D^4L^3N^2(L-N)}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h -$$

$$- \frac{D^3L^3N^2 - D^2LN(1+LN)+1}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h +$$

$$+ \frac{D^{18}L^{11}N^5 - D^{17}L^9N^5(L-1) - 2D^{16}L^9N^5(LN+1)}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h -$$

$$- \frac{D^{15}L^8N^4\big(1-N(L+1)\big) + D^{14}L^7N^4\big(1+(N+1)L-L^2N(N-1)\big)}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h -$$

$$- \frac{2D^{13}L^7N^4(L-1) + D^{12}L^6N^2\big(L^3N^3 + L(L+1)N^3 - N^2 + LN + 1\big)}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h -$$

$$- \frac{2D^{11}L^6N^3(1-L) + D^{10}L^4N^2(2L^3N^2 - L^2N^2 - L(N+1) - 1)}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h -$$

$$- \frac{D^9L^5N^2(LN+1) + D^8L^4N(LN^2+1) - D^7L^4N^2(L+1) + D^5L^3N + D^5L^2N}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h -$$

$$- \frac{D^4L^3N^2}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)} =$$

$$= \frac{D^{18}L^{11}N^5 - D^{17}L^9N^5(L-1) - 2D^{16}L^9N^5(LN+1)}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h -$$

$$- \frac{D^{15}L^8N^4\big(1-N(L+1)\big) + D^{14}L^7N^4\big(1+(N+1)L-L^2N(N-1)\big)}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h -$$

$$- \frac{2D^{13}L^7N^4(L-1) + D^{12}L^6N^2\big(L^3N^3 + L(L+1)N^3 - N^2 + LN + 1\big)}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h -$$

$$- \frac{2D^{11}L^6N^3(1-L) - D^{10}L^4N^2(L^4N^2 + 2L^3N^2 + L^2N^2 + L(N+1)+1)}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h -$$

$$-\frac{D^9L^5N^2(L^2N^2+LN+1)+D^8L^4N\left(LN^2+1+L^2N^3(LN+2)\right)}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h+$$

$$+\frac{D^7L^4N^2\left(L^5N^3(LN-1)-L+1\right)+D^6L^4N^3\left(1+(2N+1)L+L^2N\right)}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h+$$

$$+\frac{D^5L^2N\left(L^4N^3(L-1)+N+1\right)+}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}$$

$$+\frac{D^4L^3N^2(L-N-1)-D^3L^3N^2-D^2LN(1+LN)+1}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h. \quad (3.395)$$

Відповідно, чисельник передавальної функції $T_n(X_h)$, заданої співвідношенням (3.376), з урахуванням співвідношення (3.393), яке дає залежність $X_e(X_h)$, буде мати вигляд:

$$T_n(X_h)=\frac{D^4L^2NX_e}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}=$$

$$=\frac{-D^{20}L^{12}N^5+D^{19}L^{10}N^5(L-1)+2D^{18}L^{10}N^5(LN+1)}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h+$$

$$+\frac{D^{17}L^9N^4\left(1-N(L+1)\right)+D^{16}L^8N^4\left(1+(N+1)L-L^2N(N-1)\right)}{D^5L^3N^3(1-D^2L^2N)(1-D^2L^2N+D^3L)}+$$

$$+\frac{2D^{15}L^8N^4(L-1)+D^{14}L^7N^2\left(L^3N^3+L(L+1)N^3-N^2+LN+1\right)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h+$$

$$+\frac{2D^{13}L^7N^3(1-L)+D^{12}L^5N^5(2L^3N^2-L^2N^2-L(N+1)-1)}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h+$$

$$+\frac{D^{11}L^6N^5(LN+1)+D^{10}L^5N(LN^2+1)-D^9L^5N^2(L+1)+D^7L^3N(L+1)+D^7+D^6L^4N^2}{D^3L^2N^2(1-D^2L^2N)(1-D^2L^2N+D^3L)}X_h. \quad (3.396)$$

Зрозуміло, що як в чисельнику, так і в знаменнику передавальної функції для пошуку мінімальної кодової відстані нас цікавлять лише складові з мінімальною степінню D, тому решту складових для проведення подальшого аналізу можна відкинути. Велика кількість складових у передавальній функції $T(D)$ та складність цього дрібно-раціонального виразу, який ґрунтується на співвідношенні (3.376), насамперед обумовлені тим, що в отриманих співвідношеннях (3.395) та (3.396) для знаменника та чисельника передавальної функції записані всі можливі шляхи від входу до виходу

кодувальної системи, а кількість таких шляхів відповідає кількості розміщень із повтореннями для нулів та одиниць в найкоротшій кодовій комбінації. А з основ комбінаторики відомо, що кількість розміщень із повтореннями з n елементів по k визначається як $\bar{A}_n^k = n^k \gg k$ [48]. Тобто, кількість можливих кодових комбінацій значно перевищує розрядність коду, і всі можливі варіанти коду мають бути відображені в передавальній функції. Саме тому сьогодні аналіз передавальної функції згорткових кодів частіше проводиться з використанням обчислювальної техніки [33, 62, 63]. Складність постановки обчислювального завдання та його формалізації в даному разі полягає в тому, що розв'язування системи лінійних рівнянь, аналогічних (3.374), здійснюється не числовими методами, а аналітично, і узагальнених методів, які б допомогли розв'язати таку систему через формування аналітичних співвідношень, аналогічних (3.375) – (3.396), в обчислювальній математиці не існує [52, 62, 63]. Насамперед це пов'язано з тим, що зв'язки між вузлами у схемі скінченного автомату, аналогічній наведеній на рис. 3.107, зазвичай не систематизовані та мають стохастичний характер. Тому тут необхідно використовувати стандартні методи не чисельної, а дискретної математики, які безпосередньо пов'язані з комбінаторним аналізом [48, 54, 55]. Для розв'язування аналітичних завдань дискретної математики з використанням засобів комп'ютерної техніки можуть бути застосовані сучасні системи науково-технічних розрахунків із потужними символьними процесорами, зокрема MatLab або Maple [13, 14].

Аналітичний вираз для передавальної функції згорткового коду $T(D)$ суттєво спрощується лише на останньому етапі пошуку мінімальної кодової відстані, оскільки надалі, для розв'язування цього завдання, всі складові чисельника та знаменника, які мають високі степені, можуть бути відкинуті. Нас, в ході подальшого аналізу, будуть цікавити лише складові із мінімальною степінню змінної D. Тоді, враховуючи отримані, досить громіздкі аналітичні співвідношення для чисельника передавальної функції (3.396) та знаменника

(3.395), можна записати спрощений аналітичний вираз для мінімального значення функції $T(D)$, враховуючи лише складові з мінімальними степенями змінної D, наступним чином:

$$T(D) > \frac{D^6L^4N^2}{1-D^2L^2N^2} = \frac{D^6L^4N^2}{(1-DLN)(1+DLN)}. \quad (3.397)$$

Перепишемо отримане співвідношення (3.397) у вигляді степеневого ряду. Для цього спочатку проаналізуємо знаменник дрібно-раціонального виразу (3.397). Враховуючи загальновідоме співвідношення теорії поліномів:

$$1 - D^2L^2N^2 = (1 - DLN)(1 + DLN), \quad (3.398)$$

спочатку запишемо дрібно-раціональне співвідношення (3.397) у вигляді суми двох простих дробей з невідомими коефіцієнтами a та b. Відповідний аналітичний вираз має наступний вигляд:

$$T(D) = \frac{D^6L^4N^2}{(1-DLN)(1+DLN)} = \frac{aD^5L^3N}{1-DLN} + \frac{bD^5L^3N}{1+DLN} = D^5L^3N\left(\frac{a}{1-DLN} + \frac{b}{1+DLN}\right). \quad (3.399)$$

В отриманому співвідношенні (3.399) невідомі коефіцієнти a та b можна знайти, використовуючи відомий з основ дискретної математики метод невизначених коефіцієнтів [48]. Відповідно, маємо:

$$T(D) = \frac{D^6L^4N^2}{(1 - DLN)(1 + DLN)} = D^5L^3N\left(\frac{a + aDLN + b - bDLN}{(1 - DLN)(1 + DLN)}\right) =$$

$$= D^5L^3N\left(\frac{a+aDLN+b-bDLN}{(1-DLN)(1+DLN)}\right) = D^5L^3N\left(\frac{a+b-DLN(a-b)}{(1-DLN)(1+DLN)}\right). \quad (3.400)$$

Враховуючи отримане співвідношення (3.400), завдання пошуку невідомих коефіцієнтів a та b зводиться до розв'язування простої системи лінійних рівнянь:

$$\begin{cases} a + b = 0; \\ a - b = 1. \end{cases} \Rightarrow \begin{cases} 2a = 1; \\ a - b = 1. \end{cases} \Rightarrow \begin{cases} a = \frac{1}{2}; \\ b = -\frac{1}{2}. \end{cases} \quad (3.401)$$

Тоді, враховуючи значення коефіцієнтів a та b, заданих співвідношеннями (3.401), перепишемо аналітичний вираз для передавальної функції $T(D)$, заданої співвідношенням (3.400), наступним чином:

$$T(D) = \frac{D^5L^3N}{2}\left(\frac{1}{1 - DLN} - \frac{1}{1 + DLN}\right).$$

Тепер, розкладаючи отримані прості дроби в функціональний ряд з використанням співвідношення (3.365), отримуємо остаточний результат для передавальної функції $T(D)$ у вигляді степеневого ряду:

$$T(D) = \frac{D^5 L^3 N}{2}\left(\frac{1}{1 - DLN} - \frac{1}{1 + DLN}\right) =$$

$$= \frac{D^5 L^3 N}{2}(2DLN + 2D^3 L^3 N^3 + 2D^5 L^5 N^5 + \cdots) = D^6 L^4 N^2 + \cdots \qquad (3.402)$$

Тобто, згідно з остаточним виразом (3.402), проведені аналітичні перетворення дозволяють зробити наступні висновки щодо коректувальної здатності згорткового коду з параметрами $K = 4$ та $\frac{1}{n} = \frac{1}{3}$. Мінімальна кодова відстань такого згорткового коду становить $d_{\min} = 6$ і шлях за ґратковим деревом, який відповідає цій кодовій відстані, має $l = 4$ гілки, з яких дві гілки мають вагу 1, тобто є помилковими. На основі цих міркувань можна зробити головний висновок про те, що декодер згорткового коду з параметрами $K = 4$ та $\frac{1}{n} = \frac{1}{3}$, узагальнена структурна схема якого наведена на рис. 3.96, а схема відповідного скінченного автомату – на рис. 3.107, виправляє всі одиночні помилки.

Розглянемо інший приклад щодо пошуку коректувальних параметрів згорткового коду.

Приклад 3.70 Через аналіз схеми скінченного автомату, наведеної на рис. 3.70, знайти аналітичний вираз для передавальної функції декодеру згорткового коду з параметрами $K = 5$, $\frac{1}{n} = \frac{1}{3}$, узагальнена структурна схема якого наведена на рис. 3.69, та визначити мінімальну кодову відстань цього коду.

Як і в попередньому прикладі, будемо розв'язувати поставлене завдання пошуку мінімальної кодової відстані, аналізуючи схему скінченного автомату, який відповідає згортковому коду із заданими параметрами довжини кодового обмеження K та коефіцієнту надлишковості $\frac{1}{n}$. Узагальнена схема такого скінченного автомату наведена на рис. 3.70, але, як і в попередніх прикладах, для виконання завдання пошуку мінімальної кодової відстані над цією схемою

необхідно зробити відповідні структурні перетворення. По-перш за все, слід видалити петльове з'єднання на вузлі *a*, а також ввести додатковий вихідний вузол *q*, який, як і вузол *a*, відповідає стану кодеру 0000. Після цього на схемі автомату вводяться метрики гілок з параметрами *D*, *L* та *N*, і значення цих метрик відповідає вихідним кодовим комбінаціям, вказаним для схеми скінченного автомату, яка розглядається, на рис. 3.70. Схема скінченного автомату із проведеними відповідними перетвореннями його структури показана на рис. 3.108.

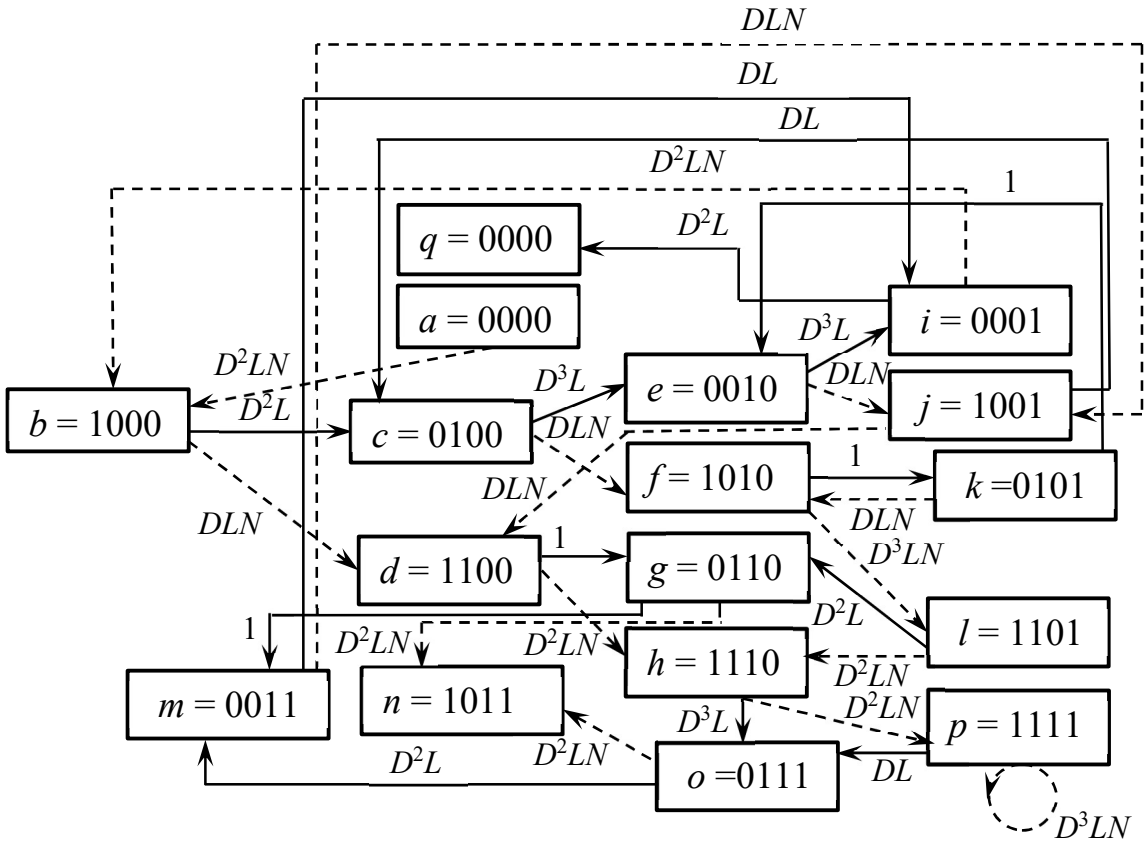


Рис. 3.108 Перетворена діаграма станів скінченного автомату для згорткового коду з параметрами $K = 5$, $\frac{1}{n} = \frac{1}{3}$, з позначеними метриками гілок, необхідними для пошуку мінімальної кодової відстані

Згідно зі схемою скінченного автомату, наведеною на рис. 3.108, систему лінійних рівнянь для пошуку величини мінімального просвіту між кодовими комбінаціями можна записати наступним чином:

$$\begin{cases} X_b = D^2LN(X_a + X_i); \\ X_c = D^2LX_b + DLX_j; \\ X_d = DLN(X_b + X_j); \\ X_e = D^3LX_c + X_k; \\ X_f = DLN(X_c + X_k); \\ X_g = X_d + D^2LX_l; \\ X_h = D^2LN(X_d + X_l); \\ X_i = D^3LX_e + DLX_m; \\ X_j = DLN(X_e + X_m); \\ X_k = X_f + D^2LX_n; \\ X_l = D^2LN(X_f + X_n); \\ X_m = X_g + D^2LX_o; \\ X_n = D^2LN(X_g + X_o); \\ X_o = D^3LX_h + DL\ _p; \\ X_p = D^2LNX_h + D^3LN\ _p; \\ X_q = D^2LX_i. \end{cases} \quad (3.403)$$

Із записаної системи рівнянь (3.403), яка відповідає схемі скінченного автомату, наведеній на рис. 3.108, можна отримати аналітичний вираз для передавальної функції $T(D)$ згорткового коду з параметрами $K = 5$ та $\frac{1}{n} = \frac{1}{3}$. Для цього спочатку необхідно записати передавальну функцію через стани системи, а потім, проводячи аналітичні викладки, записати математичний вираз для передавальної функції в іншій формі, а саме, через параметри ґраткового дерева D, L та N. Аналогічні аналітичні перетворення проводились в прикладі 3.79 та у підрозділі 3.8.13.3, проте слід зауважити, що єдиного алгоритму проведення таких перетворень не існує і для кожної схеми скінченного автомату вони є унікальними. Це насамперед пов'язано з тим, що зв'язки між станами мають стохастичний характер та закономірностей формування таких зв'язків не існує.

Спочатку запишемо аналітичний вираз для передавальної функції через вхідний та вихідний стани кодувальної системи. Відповідно, маємо:

$$T(X_a, X_q) = \frac{X_q}{X_a} = \frac{(D^2LN)D^2LX_i}{X_b - D^2LNX_i} = \frac{D^4L^2NX_i}{X_b - D^2LN\ _i}. \quad (3.404)$$

Одним із ключових для проведення подальших аналітичних перетворень є п'ятнадцяте рівняння системи (3.404), згідно з яким стан X_p залежить лише від стану X_h. Через відповідні аналітичні перетворення цього лінійного

рівняння можна отримати залежність $X_p(X_h)$ у явній формі. Відповідно, маємо:

$$X_p = D^2LN\ _h + D^3LN\ _p \Rightarrow X_p - D^3LN\ _p = D^2LN\ _h \Rightarrow X_p(X_h) = \frac{D^2LN\ _h}{1-D^3LN}. \quad (3.405)$$

З іншого боку, згідно з чотирнадцятим рівнянням системи (3.403), стан системи X_o явно залежить лише від двох станів, X_p та X_h, тобто, $X_o = X_o(X_p, X_h)$. За такої умови, враховуючи отриманий вираз (3.405), який дає безпосередню залежність $X_p(X_h)$, можна у явній формі отримати залежність між станами X_o та X_h, тобто $X_o(X_h)$. Відповідно, маємо:

$$X_o = D^3LX_h + DL\ _p = D^3LX_h + \frac{D^2LNX_h}{1-D^3LN} = \left(D^3L + \frac{D^2LN}{1-D^3LN}\right)X_h =$$

$$= \frac{-D^6L^2N+D^3L+D^2LN}{1-D^3LN}X_h. \quad (3.406)$$

Проаналізуємо тепер тринадцяте рівняння системи (3.403). Спочатку перепишемо його, з урахуванням отриманого виразу (3.406), для залежності між станами $X_o(X_h)$ наступним чином:

$$X_n = D^2LN\left(X_g + X_o\right) = D^2LN\left(X_g + \frac{-D^6L^2N+D^3L+D^2LN}{1-D^3LN}X_h\right). \quad (3.407)$$

Метою подальшого перетворення отриманого аналітичного виразу (3.407) є виключення з нього змінної X_g. Як і для інших змінних, що описують стани скінченного автомату, головною метою цих перетворень є отримання аналітичного виразу $X_n(X_h)$. Для виконання таких перетворень насамперед проаналізуємо шосте та одинадцяте рівняння системи (3.403). Після проведення підстановок та відповідних аналітичних викладок отримуємо наступний результат:

$$X_g = X_d + D^2LX_l = X_d + D^2L\left(D^2LN\left(X_f + X_n\right)\right) = X_d + D^4L^2N\left(X_f + X_n\right) =$$

$$= X_d + D^4L^2NX_f + D^4L^2NX_n. \quad (3.408)$$

Підставляючи отриманий вираз (3.408) до співвідношення (3.407), отримуємо наступний результат:

$$X_n = D^2LN\left(X_g + \frac{-D^6L^2N + D^3L + D^2LN}{1-D^3LN}X_h\right) =$$

$$= D^2LN\left(X_d + D^4L^2NX_f + D^4L^2NX_n + \frac{-D^6L^2N + D^3L + D^2LN}{1-D^3LN}X_h\right) =$$

$$= D^2LNX_d + D^6L^3N^2X_f + D^6L^3N^2X_n + \frac{-D^6L^2N + D^3L + D^2LN}{1 - D^3LN}X_h,$$

або

$$X_n(1 - D^6L^3N^2) = D^2LN\ \ _d + D^6L^3N^2X_f + \frac{-D^6L^2N + D^3L + D^2LN}{1 - D^3LN}X_h,$$

тобто

$$X_n = \frac{D^2LN}{1-D^6L^3N^2}X_d + \frac{D^6L^3N^2}{1-D^6L^3N^2}X_f + \frac{-D^6L^2N+D^3L+D^2LN}{1-D^3LN}X_h. \quad (3.409)$$

Значення стану змінної X_d, яке є необхідним для подальшого перетворення отриманого співвідношення (3.409), знайдемо, аналізуючи шосте та сьоме рівняння системи (3.403). Відповідно, маємо:

$$X_d = X_g - D^2LX_l. \quad (3.410)$$

Тоді

$$X_h = D^2LN(X_d + X_l) = D^2LN\big(X_g - D^2LX_l + X_l\big) =$$

$$= D^2LN\big(X_g + (1 - D^2L)X_l\big) = D^2LN\ \ _g + (D^2LN - D^4L^2N)X_l. \quad (3.411)$$

Отриманий вираз (3.411) перепишемо наступним чином:

$$X_g = \frac{X_h+(D^4L^2N-D^2LN)X_l}{D^2LN}. \quad (3.412)$$

Тоді, підставляючи отримане співвідношення (3.412) до аналітичного виразу (3.410), отримуємо чергову залежність між станами скінченного автомату $X_d(X_h, X_l)$:

$$X_d = \frac{X_h + (D^4L^2N - D^2LN)X_l}{D^2LN} - D^2LX_l =$$

$$= \frac{X_h + (D^4L^2N - D^2LN)X_l - D^4L^2NX_l}{D^2LN} = \frac{X_h-D^2LNX_l}{D^2LN}.$$

Тобто, остаточно, на даному етапі аналітичних перетворень, можна записати залежність між станами скінченного автомату $X_d(X_h, X_l)$ у наступному вигляді:

$$X_d = \frac{X_h-D^2LN\ \ _l}{D^2LN}. \quad (3.413)$$

Аналогічний аналітичний вираз для залежності станів скінченного автомату $X_f(X_h, X_l)$ можна отримати з одинадцятого рівняння системи (3.403). Відповідно, маємо:

$$D^2LNX_f = X_l - D^2LN\ \ _n,$$

або

$$X_f = \frac{X_l - D^2LNX_n}{D^2LN}. \tag{3.414}$$

Аналітичний вираз (3.409) для стану скінченного автомату X_n, з урахуванням отриманих залежностей $X_d(X_h, X_l)$ та $X_f(X_h, X_l)$, тобто (3.413) та (3.414), можна переписати наступним чином:

$$X_n = \frac{D^2LN}{1 - D^6L^3N^2}X_d + \frac{D^6L^3N^2}{1 - D^6L^3N^2}X_f + \frac{-D^6L^2N + D^3L + D^2LN}{1 - D^3LN}X_h =$$

$$= \left(\frac{D^2LN}{1 - D^6L^3N^2}\right)\left(\frac{X_h - D^2LN\ \ _l}{D^2LN}\right) + \left(\frac{D^6L^3N^2}{1 - D^6L^3N^2}\right)\left(\frac{X_l - D^2LNX_n}{D^2LN}\right) +$$

$$+ \frac{-D^6L^2N + D^3L + D^2LN}{1 - D^3LN}X_h = \frac{X_h - D^2LNX_l}{1 - D^6L^3N^2} + \frac{D^4L^2N(X_l - D^2LNX_n)}{1 - D^6L^3N^2} +$$

$$+ \frac{-D^6L^2N + D^3L + D^2LN}{1 - D^3LN}X_h = \frac{X_h}{1 - D^6L^3N^2} - \frac{D^2LNX_l}{1 - D^6L^3N^2} + \frac{D^4L^2NX_l}{1 - D^6L^3N^2} +$$

$$+ \frac{D^6L^3N^2X_n}{D^6L^3N^2 - 1} + \frac{-D^6L^2N + D^3L + D^2LN}{1 - D^3LN}X_h = \frac{D^4L^2N - D^2LN}{1 - D^6L^3N^2}X_l +$$

$$+ \frac{D^6L^3N^2}{D^6L^3N^2 - 1}X_n + \left(\frac{1}{1 - D^6L^3N^2} + \frac{-D^6L^2N + D^3L + D^2LN}{1 - D^3LN}\right)X_h =$$

$$= \frac{D^6L^3N^2}{D^6L^3N^2 - 1}X_n + \frac{D^4L^2N - D^2LN}{1 - D^6L^3N^2}X_l +$$

$$+ \frac{1 - D^3LN + (1 - D^6L^3N^2)(-D^6L^2N + D^3L + D^2LN)}{(1 - D^6L^3N^2)(1 - D^3LN)}X_h =$$

$$= \frac{D^6L^3N^2}{D^6L^3N^2 - 1}X_n + \frac{D^4L^2N - D^2LN}{1 - D^6L^3N^2}X_l +$$

$$+ \frac{D^{12}L^5N^3 - D^9L^4N - D^8L^4N^3 - D^6L^2N + D^3L(1-N) + D^2LN + 1}{(1 - D^6L^3N^2)(1 - D^3LN)}X_h. \tag{3.415}$$

Аналізуючи та перетворюючи отримане співвідношення (3.415), можна отримати аналітичний вираз для залежності станів скінченного автомату $X_n(X_h, X_l)$ у явному вигляді. Відповідно, маємо:

$$X_n\left(1 + \frac{D^6L^3N^2}{1 - D^6L^3N^2}\right) = \frac{D^4L^2N - D^2LN}{1 - D^6L^3N^2}X_l +$$

$$+\frac{D^{12}L^5N^3 - D^9L^4N - D^8L^4N^3 - D^6L^2N + D^3L(1-N) + D^2LN + 1}{(1 - D^6L^3N^2)(1 - D^3LN)}X_h,$$

або

$$\frac{D^6L^3N^2X_n}{1 - D^6L^3N^2} = \frac{D^4L^2N - D^2LN}{1 - D^6L^3N^2}X_l +$$
$$+\frac{D^{12}L^5N^3 - D^9L^4N - D^8L^4N^3 - D^6L^2N + D^3L(1-N) + D^2LN + 1}{(1 - D^6L^3N^2)(1 - D^3LN)}X_h,$$

тобто, остаточно:

$$X_n = \frac{D^4L^2N - D^2LN}{D^6L^3N^2}X_l +$$
$$+\frac{D^{12}L^5N^3 - D^9L^4N - D^8L^4N^3 - D^6L^2N + D^3L(1-N) + D^2LN + 1}{D^6L^3N^2(1 - D^3LN)}X_h. \quad (3.416)$$

Для пошуку аналітичної залежності між станами автомату $X_l(X_n,\ X_h)$ використаємо п'яте, десяте та одинадцяте рівняння системи (3.403). Відповідно, маємо:

$$X_l = D^2LN\left(X_f + X_n\right) = D^2LN(DLN(X_c + X_k) + X_n) =$$
$$= D^2LN\left(DLN\left(X_c + X_f + D^2LX_n\right) + X_n\right) = D^3L^2N^2X_c + D^3L^2N^2X_f +$$
$$+D^5L^3N^2X_n + D^2LN\ _n = D^3L^2N^2X_c + D^3L^2N^2X_f + D^2LN(D^3L^2N^2 + 1)X_n =$$
$$= D^3L^2N^2X_c + D^3L^2N^2\frac{X_l - D^2LN\ _n}{D^2LN} + D^2LN(D^3L^2N + 1)X_n =$$
$$= D^3L^2N^2X_c + DLN(X_l - D^2LNX_n) + D^2LN(D^3L^2N + 1)X_n =$$
$$= D^3L^2N^2X_c + DLNX_l - D^3L^2N^2X_n + D^2LN(D^3L^2N + 1)X_n =$$
$$= D^3L^2N^2X_c + DLNX_l + D^5L^3N^2X_n - D^3L^2N^2X_n + D^2LNX_n =$$
$$= D^3L^2N^2X_c + DLNX_l + (D^5L^3N^2 - D^3L^2N^2 + D^2LN)X_n.$$

або, записуючи залежність між станами скінченного автомату $X_l(X_n, X_c)$ у явній формі, маємо:

$$X_l - DLNX_l = D^3L^2N^2X_c + (D^5L^3N^2 - D^3L^2N^2 + D^2LN)X_n,$$

тобто, остаточно:

$$X_l = \frac{D^3L^2N^2X_c + \left(D^5L^3N^2 - D^3L^2N^2 + D^2LN\right)X_n}{1 - DLN}. \quad (3.417)$$

Введемо нову змінну

$$X_{nh} = \frac{D^{12}L^5N^3 - D^9L^4N - D^8L^4N^3 - D^6L^2N + D^3L(1-N) + D^2LN + 1}{D^6L^3N^2(1 - D^3LN)} X_h, \qquad (3.418)$$

та, відповідно, перепишемо аналітичний вираз $X_n(X_h, X_l)$ (3.416) у спрощеному вигляді:

$$X_n = \frac{D^4L^2N - D^2LN}{D^6L^3N^2} X_l + X_{nh}. \qquad (3.419)$$

З урахуванням отриманої залежності між станами скінченного автомату $X_l(X_n, X_c)$ (3.417) та співвідношень (3.416), (3.418), можна у явному вигляді записати залежність $X_n(X_c, X_h)$. Підставляючи аналітичний вираз $X_l(X_n, X_c)$, заданий співвідношенням (3.417), до співвідношення (3.419) та проводячи відповідні аналітичні перетворення, отримуємо наступний результат:

$$X_n = \frac{D^4L^2N - D^2LN}{D^6L^3N^2} X_l + X_{nh} =$$

$$= \left(\frac{D^4L^2N - D^2LN}{D^6L^3N^2}\right) \frac{D^3L^2N^2X_c + (D^5L^3N^2 - D^3L^2N^2 + D^2LN)X_n}{1 - DLN} + X_{nh} =$$

$$= \frac{D^4L^2N - D^2LN}{D^6L^3N^2(1 - DLN)} D^3L^2N^2X_c +$$

$$+ \frac{(D^4L^2N - D^2LN)(D^5L^3N^2 - D^3L^2N^2 + D^2LN)}{D^6L^3N^2(1 - DLN)} X_n + X_{nh} =$$

$$= \frac{D^7L^4N^3 - D^5L^3N^2}{D^6L^3N^2(1 - DLN)} X_c +$$

$$+ \frac{D^9L^5N^3 - 2D^7L^4N^3 + D^6L^3N + D^5L^3N^3 + D^4L^2N^2}{D^6L^3N^2(1 - DLN)} X_n + X_{nh} =$$

$$= \frac{D^7L^4N^3 - D^5L^3N^2}{D^6L^3N^2(1 - DLN)} X_c +$$

$$+ \frac{D^9L^5N^3 - 2D^7L^4N^3 + D^6L^3N + D^5L^3N^3 + D^4L^2N^2}{-D^7L^4N^3 + D^6L^3N^2} X_n + X_{nh},$$

або

$$\left(1 - \frac{D^9L^5N^3 - 2D^7L^4N^3 + D^6L^3N + D^5L^3N^3 + D^4L^2N^2}{-D^7L^4N^3 + D^6L^3N^2}\right) X_n =$$

$$\frac{D^9L^5N^3 - 3D^7L^4N^3 + 2D^6L^3N + D^5L^3N^3 + D^4L^2N^2}{-D^7L^4N^3 + D^6L^3N^2} X_n$$

$$= \frac{D^7L^4N^3 - D^5L^3N^2}{D^6L^3N^2(1-DLN)} X_c + X_{nh}.$$

Тоді остаточну залежність між станами скінченного автомату $X_n(X_c, X_h)$ у явній формі, з урахуванням співвідношення (3.418), можна записати у наступному вигляді:

$$X_n = \frac{D^7L^4N^3 - D^5L^3N^2}{D^9L^5N^3 - 3D^7L^4N^3 + 2D^6L^3N + D^5L^3N^3 + D^4L^2N^2} X_c +$$

$$+ \frac{D^6L^3N^2(1-DLN)}{D^9L^5N^3 - 3D^7L^4N^3 + 2D^6L^3N + D^5L^3N^3 + D^4L^2N^2} X_{nh} =$$

$$= \frac{D^7L^4N^3 - D^5L^3N^2}{D^9L^5N^3 - 3D^7L^4N^3 + 2D^6L^3N + D^5L^3N^3 + D^4L^2N^2} X_c +$$

$$+ \left(\frac{D^6L^3N^2(1-DLN)}{D^9L^5N^3 - 3D^7L^4N^3 + 2D^6L^3N + D^5L^3N^3 + D^4L^2N^2} \right) \cdot$$

$$\cdot \left(\frac{D^{12}L^5N^3 - D^9L^4N - D^8L^4N^3 - D^6L^2N + D^3L(1-N) + D^2LN + 1}{D^6L^3N^2(1-D^3LN)} \right) X_h =$$

$$= \frac{D^7L^4N^3 - D^5L^3N^2}{D^9L^5N^3 - 3D^7L^4N^3 + 2D^6L^3N + D^5L^3N^3 + D^4L^2N^2} X_c +$$

$$+ \frac{(1-DLN)(D^{12}L^5N^3 - D^9L^4N - D^8L^4N^3 - D^6L^2N + D^3L(1-N) + D^2LN + 1)}{(D^9L^5N^3 - 3D^7L^4N^3 + 2D^6L^3N + D^5L^3N^3 + D^4L^2N^2)(1-D^3LN)} X_h =$$

$$= \frac{D^7L^4N^3 - D^5L^3N^2}{D^9L^5N^3 - 3D^7L^4N^3 + 2D^6L^3N + D^5L^3N^3 + D^4L^2N^2} X_c + \frac{X_{nhn1}}{X_{nhd1}} X_h, \qquad (3.420)$$

де

$$X_{nhn1} = (1-DLN)(D^{12}L^5N^3 - D^9L^4N - D^8L^4N^3 - D^6L^2N +$$

$$D^3L(1-N) + D^2LN + 1) = D^{12}L^5N^3 - D^9L^4N - D^8L^4N^3 - D^6L^2N +$$

$$+ D^3L(1-N) + D^2LN + 1 - D^{13}L^6N^4 + D^{10}L^5N^2 + D^9L^5N^4 - D^7L^3N^2 +$$

$$+ D^4L^2N(1-N) + D^3L^2N^2 = -D^{13}L^6N^4 + D^{12}L^5N^3 + D^{10}L^5N^2 +$$

$$+ D^9L^4N(LN^3 - 1) - D^8L^4N^3 - D^7L^3N^2 - D^6L^2N + D^4L^2N(1-N) +$$

$$+ D^3\left(L^2N^2 + L(1-N)\right) + D^2LN + DLN + 1. \qquad (3.421)$$

$$X_{nhd1} = (1-D^3LN)(D^9L^5N^3 - 3D^7L^4N^3 + 2D^6L^3N + D^5L^3N^3 +$$

$$+ D^4L^2N^2) = D^9L^5N^3 - 3D^7L^4N^3 + 2D^6L^3N + D^5L^3N^3 +$$

$$+ D^4L^2N^2 - D^{12}L^6N^4 + 3D^{10}L^5N^4 - 2D^9L^4N^2 - D^8L^4N^4 -$$

$$- D^7L^3N^3 = -D^{12}L^6N^4 + 3D^{10}L^5N^4 + D^9L^4N^2(LN-2) -$$

$$-D^8L^4N^4 - D^7L^3N^3(1+3L) + 2D^6L^3N + D^5L^3N^3 + \\ +D^4L^2N^2. \quad (3.422)$$

Тепер, для подальшого пошуку залежності між двома станами скінченного автомату $X_n(X_h)$ у явній формі, необхідно, аналізуючи систему рівнянь (3.403), знайти залежність $X_c(X_h, X_n)$. Розв'язуючи друге рівняння системи (3.403) та підставляючи в нього послідовно дев'яте, четверте, дванадцяте, десяте та шосте рівняння, а також отримані раніше співвідношення між станами скінченного автомату (3.406), (3.412), (3.414) та (3.417), можна записати наступний аналітичний вираз:

$$X_c = D^2LX_b + DLX_j = D^2LX_b + D^2L^2N(X_e + X_m) = \\ = D^2LX_b + D^2L^2N\left(D^3LX_c + X_k + X_g + D^2LX_o\right) = D^2LX_b + D^5L^3NX_c + \\ +D^2L^2NX_k + D^2L^2NX_g + D^4L^3NX_o = D^2LX_b + D^5L^3NX_c + \\ +D^2L^2N\left(X_f + D^2LX_n\right) + D^2L^2N\frac{X_h + (D^4L^2N - D^2LN)X_l}{D^2LN} + \\ +D^4L^3N\left(\frac{-D^6L^2N + D^3L + D^2LN}{1 - D^3LN}X_h\right) = D^2LX_b + D^5L^3NX_c + \\ +D^2L^2N\left(\frac{X_l - D^2LN^{\ n}}{D^2LN} + D^2LX_n\right) + D^2L^2N\frac{X_h + (D^4L^2N - D^2LN)X_l}{D^2LN} + \\ +D^4L^3N\left(\frac{-D^6L^2N + D^3L + D^2LN}{1 - D^3LN}X_h\right) = D^2LX_b + D^5L^3NX_c + \\ +LX_l + D^2L(1 - LN)X_n + LX_h + (D^4L^3N - D^2L^2N)X_l + \\ +\frac{-D^{10}L^5N^2 + D^7L^4N + D^6L^4N^2}{1 - D^3LN}X_h = D^2LX_b + D^5L^3NX_c + \\ +(D^4L^3N - D^2L^2N + L)X_l + D^2L(1 - LN)X_n + \\ +\frac{-D^{10}L^5N^2 + D^7L^4N + D^6L^4N^2 - D^3L^2N + L}{1 - D^3LN}X_h = \\ = D^2LX_b + D^5L^3NX_c + D^2L(1 - LN)X_n + \\ +(D^4L^3N - D^2L^2N + L)\frac{D^3L^2N^2X_c + (D^5L^3N^2 - D^3L^2N^2 + D^2LN)X_n}{1 - DLN} + \\ +\frac{-D^{10}L^5N^2 + D^7L^4N + D^6L^4N^2 - D^3L^2N + L}{1 - D^3LN}X_h = D^2LX_b + D^5L^3NX_c +$$

$$+D^2L(1-LN)X_n+\frac{D^7L^5N^3-D^5L^4N^3+D^3L^3N^2}{1-DLN}X_c+$$
$$+\frac{D^9L^6N^3-2D^7L^5N^3+D^6L^4N^2+D^5L^4N^2(1+N)}{1-DLN}X_n+$$
$$+\frac{-D^4L^3N^2-D^3L^3N^2+D^2L^2N}{1-DLN}X_n+$$
$$+\frac{-D^{10}L^5N^2+D^7L^4N+D^6L^4N^2-D^3L^2N+L}{1-D^3LN}X_h=D^2LX_b+$$
$$+\frac{D^7L^5N^3-D^6L^4N^2+D^5L^3N(1-LN^2)+D^3L^3N^2}{1-DLN}X_c+$$
$$+\frac{D^9L^6N^3-2D^7L^5N^3+D^6L^4N^2+D^5L^4N^2(1+N)}{1-DLN}X_n+$$
$$+\frac{-D^4L^3N^2-D^3L^3N^2+D^2L^2N}{1-DLN}X_n+$$
$$+\frac{D^2L(1-LN)+D^3L^3N^2-D^4L^2N}{1-DLN}X_n+$$
$$+\frac{-D^{10}L^5N^2+D^7L^4N+D^6L^4N^2-D^3L^2N+L}{1-D^3LN}X_h=D^2LX_b+$$
$$+\frac{D^7L^5N^3-D^6L^4N^2+D^5L^3N(1-LN^2)+D^3L^3N^2}{1-DLN}X_c+$$
$$+\frac{D^9L^6N^3-2D^7L^5N^3+D^6L^4N^2+D^5L^4N^2(1+N)-D^4L^2N(1+LN)}{1-DLN}X_n+$$
$$+\frac{D^2L}{1-DL}X_n+\frac{-D^{10}L^5N^2+D^7L^4N+D^6L^4N^2-D^3L^2N+L}{1-D^3LN}X_h. \quad (3.423)$$

Зрозуміло, що отримана в процесі проведених аналітичних перетворень дрібно-раціональна функція (3.423) є досить складною та громіздкою, але остаточна залежність $X_c(X_h, X_n)$ поки ще не отримана, оскільки першою складовою рівняння (3.423) є змінна стану X_b. Тому тепер необхідно знайти аналітичну залежність між станами скінченного автомату X_b (X_c, X_h, X_n) та підставити її до отриманого проміжного співвідношення (3.423). Для цього скористаємося другим рівнянням системи (3.403) та, для спрощення аналітичних викладок, будемо відразу шукати аналітичний вираз для добутку змінних D^2LX_b. Зрозуміло, що для підстановки відповідних значень змінних стану також будуть використані інші рівняння системи (3.403), а також отримані раніше

співвідношення (3.406), (3.412), (3.414) та (3.417) для станів скінченного автомату X_o, X_g, X_f та X_l. Відповідні аналітичні перетворення мають наступний вигляд:

$$D^2LX_b = X_c - DL\big(DLN(X_e + X_m)\big) =$$

$$= X_c - D^2L^2N\big(D^3LX_c + X_k + X_g + D^2LX_o\big) = X_c - D^5L^3NX_c - D^2L^2NX_k -$$

$$-D^2L^2NX_g - D^4L^3NX_o = (1 - D^5L^3N)X_c - D^2L^2N\big(X_f + D^2LX_n\big) -$$

$$-D^2L^2NX_g - D^4L^3NX_o = (1 - D^5L^3N)X_c - D^2L^2NX_f - D^4L^3NX_n -$$

$$-D^2L^2NX_g - D^4L^3NX_o = (1 - D^5L^3N)X_c -$$

$$D^2L^2N\frac{X_l - D^2LN\ \ _n}{D^2LN} - D^4L^3NX_n - D^2L^2N\frac{X_h + (D^4L^2N - D^2LN)X_l}{D^2LN} -$$

$$-D^4L^3N\frac{-D^6L^2N + D^3L + D^2LN}{1 - D^3LN}X_h = (1 - D^5L^3N)X_c - LX_l - D^2L^2NX_n -$$

$$-D^4L^3NX_n - D^2L^2NX_h - (D^6L^4N^2 - D^4L^3N^2)X_l -$$

$$-D^4L^3N\frac{-D^6L^2N + D^3L + D^2LN}{1 - D^3LN}X_h = (1 - D^5L^3N)X_c -$$

$$D^2L^2N(1 + D^2L)X_n - (D^6L^3N^2 - D^4L^2N^2 + 1)LX_l -$$

$$-\frac{D^{10}L^5N^2 - D^7L^4N - D^6L^4N^2 + D^5L^4N^2 - D^2L^2N}{1 - D^3LN}X_h =$$

$$= (1 - D^5L^3N)X_c - D^2L^2N(1 + D^2L)X_n -$$

$$-(D^6L^3N^2 - D^4L^2N^2 + 1)L\frac{D^3L^2N^2X_c + (D^5L^3N^2 - D^3L^2N^2 + D^2LN)X_n}{1 - DLN} -$$

$$-\frac{D^{10}L^5N^2 - D^7L^4N - D^6L^4N^2 + D^5L^4N^2 - D^2L^2N}{1 - D^3LN}X_h = (-D^9L^6N^4 +$$

$$+D^7L^5N^4 - D^3L^3N^2)X_c + \frac{-D^{11}L^7N^4 + D^9L^6N^4(1 + N^2) - D^8L^5N^3}{1 - DLN}X_n +$$

$$+\frac{-D^7L^5N^4 - D^5L^4N^2 + D^3L^3N^2 - D^2L^2N}{1 - DLN}X_n -$$

$$-\frac{D^{10}L^5N^2 - D^7L^4N - D^6L^4N^2 + D^5L^4N^2 - D^2L^2N}{1 - D^3LN}X_h. \qquad (3.424)$$

Тепер можна отримати остаточну залежність між станами скінченного автомату $X_c(X_h, X_n)$, підставляючи отриманий аналітичний вираз (3.424) до

дрібно-раціональної функції (3.423) та зводячи подібні доданки. Відповідно, маємо:

$$X_c = D^2LX_b + \frac{D^7L^5N^3 - D^6L^4N^2 + D^5L^3N(1-LN^2) + D^3L^3N^2}{1-DLN}X_c +$$
$$+\frac{D^9L^6N^3 - 2D^7L^5N^3 + D^6L^4N^2 + D^5L^4N^2(1+N) - D^4L^2N(1+LN)}{1-DLN}X_n +$$
$$+\frac{D^2L}{1-DLN}X_n + \frac{-D^{10}L^5N^2 + D^7L^4N + D^6L^4N^2 - D^3L^2N + L}{1-D^3LN}X_h =$$
$$= (-D^9L^6N^4 + D^7L^5N^4 - D^3L^3N^2)X_c +$$
$$+\frac{-D^{11}L^7N^4 + D^9L^6N^4(1+N^2) - D^8L^5N^3 - D^7L^5N^4 - D^5L^4N^2}{1-DLN}X_n +$$
$$+\frac{D^3L^3N^2 - D^2L^2N}{1-DLN}X_n - \frac{D^{10}L^5N^2 - D^7L^4N - D^6L^4N^2 + D^5L^4N^2}{1-D^3LN}X_h -$$
$$-\frac{D^2L^2N}{1-D^3LN}X_h + \frac{D^7L^5N^3 - D^6L^4N^2 + D^5L^3N(1-LN^2) + D^3L^3N^2}{1-DLN}X_c +$$
$$+\frac{D^9L^6N^3 - 2D^7L^5N^3 + D^6L^4N^2 + D^5L^4N^2(1+N) - D^4L^2N(1+LN)}{1-DLN}X_n +$$
$$+\frac{D^2L}{1-DLN}X_n + \frac{-D^{10}L^5N^2 + D^7L^4N + D^6L^4N^2 - D^3L^2N + L}{1-D^3LN}X_h =$$
$$= \frac{D^{10}L^7N^5 - D^9L^6N^4 - D^8L^6N^5 + D^7L^5N^3(1+N) - D^6L^4N^2}{1-DLN}X_c +$$
$$+\frac{D^5L^3N(1-LN^2) + D^4L^4N^3}{1-DLN}X_c + \frac{-D^{11}L^7N^4 + D^9L^6N^4(1+N^2)}{1-DLN}X_n +$$
$$\frac{-D^8L^5N^3 - D^7L^5N^3(N-2) + D^6L^4N^2 + D^5L^4N^3 - D^4L^2N(1+LN)}{1-DLN}X_n +$$
$$+\frac{D^3L^3N^2 + D^2L(1-LN)}{1-DLN}X_n + \frac{-2D^{10}L^5N^2 + 2D^7L^4N + 2D^6L^4N^2}{1-D^3LN}X_h +$$
$$+\frac{-D^5L^4N^2 - D^3L^2N + D^2L^2N + L}{1-D^3LN}X_h. \qquad (3.425)$$

Тепер, для отримання аналітичної залежності між станами скінченного автомату у явному вигляді, як функцію $X_c(X_h, X_n)$, необхідно переписати співвідношення (3.425) наступним чином: доданок, який відповідає стану X_c, перенести до лівої частини рівняння, звісти у цій частині рівняння подібні

доданки, а потім розділити всю праву частину рівняння на дрібну-раціональну функцію, яка стоятиме як множник перед змінною X_c. Відповіді аналітичні перетворення мають наступний вигляд:

$$X_c - \frac{D^{10}L^7N^5 - D^9L^6N^4 - D^8L^6N^5 + D^7L^5N^3(1+N) - D^6L^4N^2}{1-DLN}X_c -$$

$$-\frac{D^5L^3N(1-LN^2) + D^4L^4N^3}{1-DLN}X_c = \frac{-D^{10}L^7N^5 + D^9L^6N^4 + D^8L^6N^5}{1-DLN}X_c +$$

$$\frac{-D^7L^5N^3(1+N) + D^6L^4N^2 + D^5L^3N(1-LN^2) + D^4L^4N^3 - DLN + 1}{1-DLN}X_c =$$

$$= \frac{-D^{11}L^7N^4 + D^9L^6N^4(1+N^2) - D^8L^5N^3 - D^7L^5N^3(N-2) + D^6L^4N^2}{1-DLN}X_n +$$

$$+\frac{D^5L^4N^3 - D^4L^2N(1+LN) + D^3L^3N^2 + D^2L(1-LN)}{1-DLN}X_n +$$

$$+\frac{-2D^{10}L^5N^2 + 2D^7L^4N + 2D^6L^4N^2 - D^5L^4N^2 - D^3L^2N}{1-D^3LN}X_h +$$

$$+\frac{D^2L^2N+L}{1-D^3LN}X_h. \qquad (3.426)$$

Для отримання явної функціональної залежності $X_c(X_h, X_n)$, з метою полегшення подальших аналітичних перетворень та запису відповідних дрібно-раціональних функцій, введемо наступну додаткову змінну:

$$X_{cnhd} = -D^{10}L^7N^5 + D^9L^6N^4 + D^8L^6N^5 - D^7L^5N^3(1+N) + D^6L^4N^2 +$$

$$+D^5L^3N(1-LN^2) + D^4L^4N^3 - DLN + 1. \qquad (3.427)$$

Підставляючи введену змінну X_{cnhd}, задану поліноміальною функцією (3.427) відносно змінної D, до отриманого рівняння (3.426), маємо наступний результат для змінної стану скінченного автомату X_c:

$$X_c(X_h, X_n) = \frac{-D^{11}L^7N^4 + D^9L^6N^4(1+N^2) - D^8L^5N^3 - D^7L^5N^3(N-2)}{X_{cnhd}}X_n +$$

$$+\frac{D^6L^4N^2 + D^5L^4N^3 - D^4L^2N(1+LN) + D^3L^3N^2 + D^2L(1-LN)}{X_{cnhd}}X_n +$$

$$+\frac{(1-DLN)(-2D^{10}L^5N^2 + 2D^7L^4N + 2D^6L^4N^2 - D^5L^4N^2 - D^3L^2N)}{X_{cnhd}(1-D^3LN)}X_h +$$

$$+\frac{(1-DLN)(D^2L^2N+L)}{X_{cnhd}(1-D^3LN)}X_h=\frac{-D^{11}L^7N^4+D^9L^6N^4(1+N^2)-D^8L^5N^3}{X_{cnhd}}X_n+$$

$$+\frac{-D^7L^5N^3(N-2)+D^6L^4N^2+D^5L^4N^3-D^4L^2N(1+LN)+D^3L^3N^2}{X_{cnhd}}X_n+$$

$$+\frac{D^2L(1-LN)}{X_{cnhd}}X_n+\frac{2D^{11}L^6N^3-2D^{10}L^5N^2-2D^8L^5N^2}{(1-D^3LN)X_{cnhd}}X_h+$$

$$+\frac{2D^7L^4N(1-LN^2)+D^6L^4N^2(LN+2)-D^5L^4N^2+D^4L^3N^2}{(1-D^3LN)X_{cnhd}}X_h+$$

$$+\frac{D^3L^2N(1-LN)+D^2L^2N-DL^2N+L}{(1-D^3LN)X_{cnhd}}X_h. \quad (3.428)$$

Маючи аналітичну залежність між станами скінченного автомату $X_c(X_h, X_n)$, задану співвідношеннями (3.427), (3.428) та підставляючи функцію $X_c(X_h, X_n)$ до отриманого раніше співвідношення (3.420), яке дає функціональну залежність $X_n(X_c, X_h)$, отримуємо остаточну форму явної аналітичної залежності між двома станами скінченного $X_n(X_h)$.

Для спрощення подальшого проведення таких аналітичних перетворень з дрібно-раціональними виразами співвідношень (3.420), (3.428) та для більш компактного запису результатів аналітичних розрахунків спочатку введемо ще дві нові змінні, які мають відповідати громіздким чисельникам дробей, записаних у співвідношенні (3.428):

$$X_{cnn}=-D^{11}L^7N^4+D^9L^6N^4(1+N^2)-D^8L^5N^3-D^7L^5N^3(N-2)+$$

$$+D^6L^4N^2+D^5L^4N^3-D^4L^2N(1+LN)+D^3L^3N^2+D^2L(1-LN), \quad (3.429)$$

$$X_{chn}=2D^{11}L^6N^3-2D^{10}L^5N^2-2D^8L^5N^2+2D^7L^4N(1-LN^2)+$$

$$+D^6L^4N^2(LN+2)-D^5L^4N^2+D^4L^3N^2+D^3L^2N(1-LN)+D^2L^2N-$$

$$-DL^2N+L. \quad (3.430)$$

З урахуванням введених додаткових змінних та співвідношень (3.429), (3.430), залежність між станами скінченного автомату $X_c(X_h, X_n)$, задану співвідношенням (3.428), можна переписати у прощеному вигляді наступним чином:

$$X_c(X_h, X_n) = \frac{X_{cnn}}{X_{cnhd}} X_n + \frac{X_{chn}}{(1-D^3LN)X_{cnhd}} X_h. \quad (3.431)$$

Враховуючи спрощений вигляд функціональної залежності між станами скінченного автомату $X_c(X_h, X_n)$, заданої співвідношенням (3.431), а також записані аналітичні вирази (3.427), (3.429), (3.430) для додаткових змінних X_{cnn}, X_{chn} та X_{cnhd}, можна переписати аналітичну залежність між станами скінченного автомату $X_n(X_c, X_h)$, задану співвідношенням (3.420), наступним чином:

$$X_n = \frac{D^7L^4N^3 - D^5L^3N^2}{D^9L^5N^3 - 3D^7L^4N^3 + 2D^6L^3N + D^5L^3N^3 + D^4L^2N^2} X_c + \frac{X_{nhn1}}{X_{nhd1}} X_h =$$

$$= \frac{D^7L^4N^3 - D^5L^3N^2}{D^9L^5N^3 - 3D^7L^4N^3 + 2D^6L^3N + D^5L^3N^3 + D^4L^2N^2} \left(\frac{X_{cnn}}{X_{cnhd}} X_n + \right.$$

$$\left. + \frac{X_{chn}}{(1 - D^3LN)X_{cnhd}} X_h \right) + \frac{X_{nhn1}}{X_{nhd1}} X_h =$$

$$= \frac{(D^7L^4N^3 - D^5L^3N^2)X_{cnn}}{(D^9L^5N^3 - 3D^7L^4N^3 + 2D^6L^3N + D^5L^3N^3 + D^4L^2N^2)X_{cnhd}} X_n +$$

$$+ \left(\frac{X_{chn}}{(1-D^3LN)X_{cnhd}} + \frac{X_{nhn1}}{X_{nhd1}} \right) X_h. \quad (3.432)$$

Розглянемо кожну із складових дрібно-раціональних виразів отриманого співвідношення (3.432), аналізуючи окремо чисельники та знаменники дробей для спрощення запису відповідних математичних виразів. Для чисельника першого доданку співвідношення (3.432), враховуючи аналітичний вираз для додаткової змінної X_{cnn}, заданий співвідношенням (3.429), можна записати:

$$X_{nn1} = (D^7L^4N^3 - D^5L^3N^2)X_{cnn} = (D^7L^4N^3 - D^5L^3N^2)(-D^{11}L^7N^4 +$$

$$+D^9L^6N^4(1 + N^2) - D^8L^5N^3 - D^7L^5N^3(N - 2) + D^6L^4N^2 + D^5L^4N^3 -$$

$$-D^4L^2N(1 + LN) + D^3L^3N^2 + D^2L(1 - LN) = -D^{18}L^{11}N^7 +$$

$$+D^{16}L^{10}N^7(1 + N^2) - D^{15}L^9N^6 - D^{14}L^9N^6(N - 2) + D^{13}L^8N^5 + D^{12}L^8N^6 -$$

$$-D^{11}L^6N^4(1 + LN) + D^{10}L^7N^5 + D^9L^5N^3(1 - LN) + D^{16}L^{10}N^6 -$$

$$-D^{14}L^9N^7(1 + N^2) + D^{13}L^8N^5 + D^{12}L^8N^5(N - 2) - D^{11}L^7N^4 - D^{10}L^7N^5 +$$

$$+D^9L^5N^3(1 + LN) - D^8L^6N^4 - D^7L^4N^2(1 - LN) = -D^{18}L^{11}N^7 +$$

$$+D^{16}L^{10}N^7(1+N^2)+D^{16}L^{10}N^6-D^{15}L^9N^6-D^{14}L^9N^6(N-2)-$$
$$-D^{14}L^9N^7(1+N^2)+D^{13}L^8N^5+D^{13}L^8N^5+D^{12}L^8N^6+D^{12}L^8N^5(N-2)-$$
$$-D^{11}L^6N^4(1+LN)-D^{11}L^7N^4+D^{10}L^7N^5-D^{10}L^7N^5+D^9L^5N^3(1-LN)+$$
$$+D^9L^5N^3(1+LN)-D^8L^6N^4-D^7L^4N^2(1-LN)=-D^{18}L^{11}N^7+$$
$$+D^{16}L^{10}N^6(1+N+N^3)-D^{15}L^9N^6-D^{14}L^9N^6(N^3+2N-2)+D^{13}L^8N^5+$$
$$+2D^{12}L^8N^5(N-1)-D^{11}L^6N^4(L(N+1)+1)+2D^9L^5N^3(1-LN)-$$
$$-D^8L^6N^4-D^7L^4N^2(1-LN). \qquad (3.433)$$

Аналогічно, аналітичний вираз для знаменника першого доданку співвідношення (3.432), з урахуванням співвідношення (3.427), можна записати наступним чином:

$$X_{nd}=(D^9L^5N^3-3D^7L^4N^3+2D^6L^3N+D^5L^3N^3+D^4L^2N^2)X_{cnhd}=$$
$$=(D^9L^5N^3-3D^7L^4N^3+2D^6L^3N+D^5L^3N^3+D^4L^2N^2)(-D^{10}L^7N^5+$$
$$+D^9L^6N^4+D^8L^6N^5-D^7L^5N^3(1+N)+D^6L^4N^2+D^5L^3N(1-LN^2)+$$
$$+D^4L^4N^3-DLN+1)=-D^{19}L^{12}N^8+D^{18}L^{11}N^7+D^{17}L^{11}N^8-$$
$$-D^{16}L^{10}N^6(1+N)+D^{15}L^9N^5+D^{14}L^8N^4(1-LN^2)+D^{13}L^9N^6-$$
$$-D^{10}L^6N^4+D^9L^5N^3+3D^{17}L^{11}N^8-3D^{16}L^{10}N^7-3D^{15}L^{10}N^8+$$
$$+3D^{14}L^9N^6(1+N)-3D^{13}L^8N^5-3D^{12}L^7N^4(1-LN^2)+$$
$$+3D^{11}L^8N^6+3D^8L^5N^4-3D^7L^4N^3-2D^{16}L^{10}N^6+2D^{15}L^9N^5+$$
$$+2D^{14}L^9N^6-2D^{13}L^8N^4(1+N)+2D^{12}L^7N^3+2D^{11}L^6N^2(1-LN^2)+$$
$$+2D^{10}L^7N^4-2D^7L^4N^2+2D^6L^3N-D^{15}L^{10}N^8+D^{14}L^9N^7+D^{13}L^9N^8-$$
$$-D^{12}L^8N^6(1+N)+D^{11}L^7N^5+D^{10}L^6N^4(1-LN^2)+D^9L^7N^6-D^6L^4N^4+$$
$$+D^5L^3N^3-D^{14}L^9N^7+D^{13}L^8N^6+D^{12}L^8N^7-D^{11}L^7N^5(1+N)+$$
$$+D^{10}L^6N^4+D^9L^5N^3(1-LN^2)+D^8L^6N^5-D^5L^3N^3+D^4L^2N^2=$$
$$=-D^{19}L^{12}N^8+D^{18}L^{11}N^7+D^{17}L^{11}N^8+3D^{17}L^{11}N^8-3D^{16}L^{10}N^7-$$
$$-D^{16}L^{10}N^6(1+N)-2D^{16}L^{10}N^6+2D^{15}L^9N^5-3D^{15}L^{10}N^8+D^{15}L^9N^5-$$
$$-D^{15}L^{10}N^8+D^{14}L^8N^4(1-LN^2)+3D^{14}L^9N^6(1+N)+2D^{14}L^9N^6-D^{14}L^9N^7+$$
$$+D^{14}L^9N^7+D^{13}L^9N^6-3D^{13}L^8N^5-2D^{13}L^8N^4(1+N)+D^{13}L^8N^6+$$

$$+D^{13}L^9N^8-3D^{12}L^7N^4(1-LN^2)+2D^{12}L^7N^3-D^{12}L^8N^6(1+N)+$$
$$+D^{12}L^8N^7+3D^{11}L^8N^6+2D^{11}L^6N^2(1-LN^2)+D^{11}L^7N^5-$$
$$-D^{11}L^7N^5(1+N)-D^{10}L^6N^4+2D^{10}L^7N^4+D^{10}L^6N^4(1-LN^2)+D^{10}L^6N^4+$$
$$+D^9L^5N^3+D^9L^5N^3(1-LN^2)+D^9L^7N^6+3D^8L^5N^4+D^8L^6N^5-3D^7L^4N^3-$$
$$-2D^7L^4N^2+2D^6L^3N-D^6L^4N^4+D^5L^3N^3-D^5L^3N^3+D^4L^2N^2=$$
$$=-D^{19}L^{12}N^8+D^{18}L^{11}N^7+4D^{17}L^{11}N^8-D^{16}L^{10}N^6(4N+3)+$$
$$+D^{15}L^9N^5(3-4LN^3)+D^{14}L^8N^4\left(1+LN^2(3N+4)\right)+$$
$$+D^{13}L^8N^4(LN^4+(L+1)N^2-5N-2)+D^{12}L^7N^3(2(1+LN^3)-3N)+$$
$$+D^{11}L^6N^2\left(LN^4(3L-1)+2(1-LN^2)\right)+D^{10}L^7N^4(3-LN^2)+$$
$$+D^9L^5N^3(L^2N^3-LN^2+2)+D^8L^5N^4(LN+3)-D^7L^4N^2(3N+2)+$$
$$+D^6L^3N(2-LN^3)+D^4L^2N^2. \qquad (3.434)$$

З урахуванням отриманих співвідношень (3.433), (3.434), перепишемо аналітичний вираз (3.432) для функціональної залежності між станами скінченного автомату $X_n(X_h)$ у явній формі. Для цього необхідно перенести доданок, який відповідає стану X_n, у ліву частину рівняння, звести подібні доданки, а потім розділити праву частину рівняння на множник, який стоїть перед змінною X_n. Результат таких аналітичних перетворень для співвідношення (3.432), з урахуванням введених додаткових змінних X_{nd1} та X_{nd2}, заданих співвідношеннями (3.433) та (3.433), є досить простим та його можна записати наступним чином:

$$X_n=\left(\frac{X_{chn}}{(1-D^3LN)X_{cnhd}}+\frac{X_{nhn1}}{X_{nhd1}}\right)\left(\frac{X_{nd}}{X_{nd}\ -X_{nn1}}\right)X_h. \qquad (3.435)$$

Знаменник дробу, записаного у других дужках, також перепишемо в поліноміальній формі:

$$X_{nd}\ =X_{nd}\ -X_{nn}\ =-D^{19}L^{12}N^8+D^{18}L^{11}N^7+4D^{17}L^{11}N^8-$$
$$-D^{16}L^{10}N^6(4N+3)+D^{15}L^9N^5(3-4LN^3)+D^{14}L^8N^4\left(1+LN^2(3N+4)\right)+$$
$$+D^{13}L^8N^4(LN^4+(L+1)N^2-5N-2)+D^{12}L^7N^3(2(1+LN^3)-3N)+$$
$$+D^{11}L^6N^2\left(LN^4(3L-1)+2(1-LN^2)\right)+D^{10}L^7N^4(3-LN^2)+$$

$$+D^9L^5N^3(L^2N^3-LN^2+2)+D^8L^5N^4(LN+3)-D^7L^4N^2(3N+2)+$$
$$+D^6L^3N(2-LN^3)+D^4L^2N^2+D^{18}L^{11}N^7-D^{16}L^{10}N^6(1+N+N^3)+$$
$$+D^{15}L^9N^6+D^{14}L^9N^6(N^3+2N-2)-D^{13}L^8N^5-$$
$$-2D^{12}L^8N^5(N-1)+D^{11}L^6N^4(L(N+1)+1)-2D^9L^5N^3(1-LN)+$$
$$+D^8L^6N^4+D^7L^4N^2(1-LN)=-D^{19}L^{12}N^8+D^{18}L^{11}N^7+D^{18}L^{11}N^7+$$
$$+4D^{17}L^{11}N^8-D^{16}L^{10}N^6(4N+3)-D^{16}L^{10}N^6(1+N+N^3)+$$
$$+D^{15}L^9N^5(3-4LN^3)+D^{15}L^9N^6+D^{14}L^8N^4\left(1+LN^2(3N+4)\right)+$$
$$+D^{14}L^9N^6(N^3+2N-2)-D^{13}L^8N^5+$$
$$+D^{13}L^8N^4(LN^4+(L+1)N^2-5N-2)+D^{12}L^7N^3(2(1+LN^3)-3N)-$$
$$-2D^{12}L^8N^5(N-1)+D^{11}L^6N^2\left(LN^4(3L-1)+2(1-LN^2)\right)+$$
$$+D^{11}L^6N^4(L(N+1)+1)+D^{10}L^7N^4(3-LN^2)+$$
$$+D^9L^5N^3(L^2N^3-LN^2+2)-2D^9L^5N^3(1-LN)+$$
$$+D^8L^5N^4(LN+3)+D^8L^6N^4+D^7L^4N^2(1-LN)-D^7L^4N^2(3N+2)+$$
$$+D^6L^3N(2-LN^3)+D^4L^2N^2=-D^{19}L^{12}N^8+2D^{18}L^{11}N^7+4D^{17}L^{11}N^8-$$
$$-D^{16}L^{10}N^6(N^3+5N+4)+D^{15}L^9N^5(3+N-4LN^3)+$$
$$+D^{14}L^8N^4(LN^5+5LN^3+2LN^2+1)+$$
$$+D^{13}L^8N^4(LN^4+(L+1)N^2-6N-2)+D^{12}L^7N^3(3LN^3-LN^2-3N+2)+$$
$$+D^{11}L^6N^2(LN^4(3L-1)+N^2(NL-L+1)+2)+D^{10}L^7N^4(3-LN^2)+$$
$$+D^9L^6N^4(LN^2-N+2)+D^8L^5N^4(L(N+1)+3)$$
$$+D^7L^4N^2(N(L+3)+1)+D^6L^3N(2-LN^3)+D^4L^2N^2. \qquad (3.436)$$

Тепер необхідно проаналізувати множник аналітичного виразу (3.435), який стоїть в перших дужках та звести його до єдиного дрібно-раціонального виразу, використовуючи аналітичні поліноміальні співвідношення для додаткових змінних X_{chn}, X_{cnhd}, X_{nhn1} та X_{nhn2}, задані співвідношеннями (3.421), (3.422), (3.427) та (3.430). Спочатку перепишімо суму дробей, яка стоїть у перших дужках співвідношення (3.435), у вигляді єдиного дробу. Відповідно, маємо:

$$X_{nh1}=\frac{X_{chn}}{(1-D^3LN)X_{cnhd}}+\frac{X_{nhn1}}{X_{nhd1}}=\frac{X_{chn}X_{nhd12}+X_{nhn1}X_{cnhd}}{(1-D^3LN)X_{cnhd}X_{nhd12}}, \qquad (3.437)$$

де

$$X_{nhd12} = D^9L^5N^3 - 3D^7L^4N^3 + 2D^6L^3N + D^5L^3N^3 + D^4L^2N^2. \quad (3.438)$$

Аналізуючи аналітичній вираз (3.437) з урахуванням співвідношень співвідношеннями (3.421), (3.422), (3.427) (3.430) та (3.438), отримуємо наступні поліноміальні залежності як функції від параметрів згортоквого коду *D*, *L* та *N*:

$$X_{nn2} = X_{chn}X_{nhd12} = (2D^{11}L^6N^3 - 2D^{10}L^5N^2 - 2D^8L^5N^2 +$$
$$+2D^7L^4N(1 - LN^2) + D^6L^4N^2(LN + 2) - D^5L^4N^2 + D^4L^3N^2 +$$
$$+D^3L^2N(1 - LN) + D^2L^2N - DL^2N + L)(D^9L^5N^3 - 3D^7L^4N^3 +$$
$$+2D^6L^3N + D^5L^3N^3 + D^4L^2N^2) = 2D^{20}L^{11}N^6 - 2D^{19}L^{10}N^5 - 2D^{17}L^{10}N^5 +$$
$$+2D^{16}L^9N^4(1 - LN^2) + D^{15}L^9N^5(LN + 2) - D^{14}L^9N^5 + D^{13}L^8N^5 +$$
$$+D^{12}L^7N^4(1 - LN) + D^{11}L^7N^4 - D^{10}L^7N^5 + D^9L^6N^3 -$$
$$-6D^{18}L^{10}N^6 + 6D^{17}L^{11}N^5 + 6D^{15}L^9N^5 - 6D^{14}L^8N^4(1 - LN^2) -$$
$$-3D^{13}L^8N^5(LN + 2) + 3D^{12}L^8N^5 - 3D^{11}L^7N^5 - 3D^{10}L^6N^4(1 - LN) -$$
$$-3D^9L^6N^4 + 3D^8L^6N^4 - 3D^7L^5N^3 + 4D^{17}L^9N^4 - 4D^{16}L^8N^3 -$$
$$-4D^{14}L^8N^3 + 4D^{13}L^7N^2(1 - LN^2) + 2D^{12}L^7N^3(LN + 2) -$$
$$-2D^{11}L^7N^3 + 2D^{10}L^6N^3 + 2D^9L^5N^2(1 - LN) + 2D^8L^5N^2 -$$
$$-2D^7L^5N^2 + 2D^6L^4N + 2D^{16}L^9N^6 - 2D^{15}L^8N^5 - 2D^{13}L^8N^5 +$$
$$+2D^{12}L^7N^4(1 - LN^2) + D^{11}L^7N^5(LN + 2) - D^{10}L^7N^5 + D^9L^6N^5 -$$
$$+D^8L^5N^4(1 - LN) + D^7L^5N^4 - D^6L^5N^4 + D^5L^4N^3 + 2D^{15}L^8N^5 -$$
$$-2D^{14}L^7N^4 - 2D^{12}L^7N^4 + +2D^{11}L^6N^3(1 - LN^2) + D^{10}L^6N^4(LN + 2) -$$
$$-D^9L^6N^4 + D^7L^5N^4 + D^7L^4N^4(1 - LN) + D^6L^4N^4 - D^5L^4N^3 + D^4L^3N^2 =$$
$$= 2D^{20}L^{11}N^6 - 2D^{19}L^{10}N^5 - 6D^{18}L^{10}N^6 + 6D^{17}L^{11}N^5 - 2D^{17}L^{10}N^5 +$$
$$+4D^{17}L^9N^4 + 2D^{16}L^9N^4(1 - LN^2) - 4D^{16}L^8N^3 + 2D^{16}L^9N^6 -$$
$$-2D^{15}L^8N^5 + D^{15}L^9N^5(LN + 2) + 2D^{15}L^8N^5 + 6D^{15}L^9N^5 -$$
$$-D^{14}L^9N^5 - 2D^{14}L^7N^4 - 6D^{14}L^8N^4(1 - LN^2) - 4D^{14}L^8N^3 + D^{13}L^8N^5 -$$
$$+4D^{13}L^7N^2(1 - LN^2) - 3D^{13}L^8N^5(LN + 2) - 2D^{13}L^8N^5 +$$
$$+2D^{12}L^7N^3(LN + 2) + D^{12}L^7N^4(1 - LN) + 3D^{12}L^8N^5 -$$
$$+2D^{12}L^7N^4(1 - LN^2) - 2D^{12}L^7N^4 - 3D^{11}L^7N^5 + D^{11}L^7N^4 - 2D^{11}L^7N^3 +$$
$$+D^{11}L^7N^5(LN + 2) + 2D^{11}L^6N^3(1 - LN^2) - D^{10}L^7N^5 + 2D^{10}L^6N^3 -$$
$$+D^{10}L^6N^4(LN + 2) - 3D^{10}L^6N^4(1 - LN) - 3D^9L^6N^4 + D^9L^6N^3 + D^9L^6N^5 +$$

$$+2D^9L^5N^2(1-LN)-D^9L^6N^4+D^8L^5N^4(1-LN)+2D^8L^5N^2-3D^8L^6N^4-$$
$$-3D^7L^5N^3-2D^7L^5N^2+D^7L^5N^4+D^7L^5N^4+D^7L^4N^4(1-LN)+$$
$$+2D^6L^4N+D^4L^3N^2=2D^{20}L^{11}N^6-2D^{19}L^{10}N^5-6D^{18}L^{10}N^6+6D^{17}L^{11}N^5-$$
$$-2D^{17}L^{10}N^5+4D^{17}L^9N^4+2D^{16}L^9N^4(1-LN^2)-4D^{16}L^8N^3+2D^{16}L^9N^6-$$
$$-2D^{15}L^8N^5+D^{15}L^9N^5(LN+2)+2D^{15}L^8N^5+6D^{15}L^9N^5-D^{14}L^9N^5-$$
$$-2D^{14}L^7N^4-6D^{14}L^8N^4(1-LN^2)-4D^{14}L^8N^3+D^{13}L^8N^5-$$
$$+4D^{13}L^7N^2(1-LN^2)-3D^{13}L^8N^5(LN+2)-2D^{13}L^8N^5+$$
$$+2D^{12}L^7N^3(LN+2)+D^{12}L^7N^4(1-LN)+3D^{12}L^8N^5+$$
$$+2D^{12}L^7N^4(1-LN^2)-2D^{12}L^7N^4-3D^{11}L^7N^5+D^{11}L^7N^4-2D^{11}L^7N^3+$$
$$+D^{11}L^7N^5(LN+2)+2D^{11}L^6N^3(1-LN^2)-D^{10}L^7N^5+2D^{10}L^6N^3-$$
$$+D^{10}L^6N^4(LN+2)-3D^{10}L^6N^4(1-LN)-3D^9L^6N^4+D^9L^6N^3+D^9L^6N^5+$$
$$+2D^9L^5N^2(1-LN)-D^9L^6N^4+D^8L^5N^4(1-LN)+2D^8L^5N^2-3D^8L^6N^4-$$
$$-3D^7L^5N^3-2D^7L^5N^2+D^7L^5N^4+D^7L^5N^4+D^7L^4N^4(1-LN)+$$
$$+2D^6L^4N+D^4L^3N^2=2D^{20}L^{11}N^6-2D^{19}L^{10}N^5-6D^{18}L^{10}N^6+$$
$$+2D^{17}L^9N^4(3L^2N-LN+2)+2D^{16}L^8N^3\big(LN\big(1+N^2(1-L)\big)-2\big)+$$
$$+D^{15}L^9N^5(LN+8)-D^{14}L^7N^3(L^2N^2+6LN(1-LN^2)+2N+4)+$$
$$+D^{13}L^7N^2\big(4(1-LN^2)-LN^3(3LN+4)\big)+$$
$$+D^{12}L^7N^3\big(LN^2(2-N)+2(LN+2)\big)+$$
$$+D^{11}L^6N^3\big(L(N^2(LN-1)+N-2)+2(1-LN^2)\big)+$$
$$+D^{10}L^6N^3(-3LN^2-N+2)+D^9L^5N^2(LN(N^2-4N)-LN+2)+$$
$$+D^8L^5N^2\big(N^2\big(1-L(N+3)\big)+2\big)+D^7L^5N^2(N^2(3-LN)-3N+2)+$$
$$+2D^6L^4N+D^4L^3N^2. \qquad (3.439)$$

Аналогічно, для другого доданку в чисельнику співвідношення (3.437) можна записати наступний аналітичний вираз:

$$X_{nn3}=X_{nhn1}X_{cnhd}=(-D^{13}L^6N^4+D^{12}L^5N^3+D^{10}L^5N^2+$$
$$+D^9L^4N(LN^3-1)-D^8L^4N^3-D^7L^3N^2-D^6L^2N+D^4L^2N(1-N)+$$
$$+D^3\big(L^2N^2+L(1-N)\big)+D^2LN+DLN+1\big)(-D^{10}L^7N^5+D^9L^6N^4+$$
$$+D^8L^6N^5-D^7L^5N^3(1+N)+D^6L^4N^2+D^5L^3N(1-LN^2)+$$
$$+D^4L^4N^3-DLN+1)=D^{23}L^{13}N^9-D^{22}L^{12}N^8-D^{20}L^{12}N^7-$$

$$-D^{19}L^{11}N^6(LN^3-1)+D^{18}L^{11}N^8+D^{17}L^{10}N^7+D^{16}L^9N^6-$$
$$-D^{14}L^9N^6(1-N)-D^{13}L^7N^5\left(L^2N^2+L(1-N)\right)-D^{12}L^8N^6-D^{11}L^8N^6-$$
$$-D^{10}L^7N^5-D^{22}L^{12}N^8+D^{21}L^{11}N^7+D^{19}L^{11}N^6+D^{18}L^{10}N^5(LN^3-1)-$$
$$-D^{17}L^{10}N^7-D^{16}L^9N^6-D^{15}L^8N^5+D^{13}L^8N^5(1-N)+$$
$$+D^{12}L^6N^4\left(L^2N^2+L(1-N)\right)+D^{11}L^7N^5+D^{10}L^7N^5+D^9L^6N^4-$$
$$-D^{21}L^{12}N^9+D^{20}L^{11}N^8+D^{18}L^{11}N^7+D^{17}L^{10}N^6(LN^3-1)-$$
$$-D^{16}L^{10}N^8-D^{15}L^9N^7-D^{14}L^8N^6+D^{12}L^8N^6(1-N)+$$
$$+D^{11}L^6N^5\left(L^2N^2+L(1-N)\right)+D^{10}L^7N^6+D^9L^7N^6+D^8L^6N^5+$$
$$+D^{20}L^{11}N^7(N+1)-D^{19}L^{10}N^6(N+1)-D^{17}L^{10}N^5(N+1)-$$
$$-D^{16}L^9N^4(N+1)(LN^3-1)+D^{15}L^9N^6(N+1)+D^{14}L^8N^5(N+1)+$$
$$+D^{13}L^7N^4(N+1)-D^{11}L^7N^4(1-N^2)-$$
$$-D^{10}L^5N^3(N+1)\left(L^2N^2+L(1-N)\right)-D^9L^6N^4-D^8L^6N^4-D^7L^5N^3-$$
$$-D^{19}L^{10}N^6+D^{18}L^9N^5+D^{16}L^9N^4+D^{15}L^8N^3(LN^3-1)-D^{14}L^8N^5-$$
$$-D^{13}L^7N^4-D^{12}L^6N^3+D^{10}L^6N^3(1-N)+D^9L^4N^2\left(L^2N^2+L(1-N)\right)+$$
$$+D^8L^5N^3+D^7L^5N^3+D^6L^4N^2-D^{18}L^9N^5(1-LN^2)+D^{17}L^8N^4(1-LN^2)+$$
$$+D^{15}L^8N^3(1-LN^2)+D^{14}L^7N^2(LN^3-1)(1-LN^2)-D^{13}L^7N^4(1-LN^2)-$$
$$-D^{12}L^6N^3(1-LN^2)-D^{11}L^5N^2(1-LN^2)+D^9L^5N^2(1-N)(1-LN^2)+$$
$$+D^8L^3N\left(L^2N^2+L(1-N)\right)(1-LN^2)+D^7L^4N^2(1-LN^2)+$$
$$+D^6L^4N^2(1-LN^2)+D^5L^3N(1-LN^2)-D^{17}L^{10}N^7+D^{16}L^9N^6+D^{14}L^9N^5+$$
$$+D^{13}L^6N^4(LN^3-1)-D^{12}L^8N^6-D^{11}L^7N^5-D^{10}L^6N^4+D^8L^6N^4(1-N)+$$
$$+D^7L^4N^3\left(L^2N^2+L(1-N)\right)+D^6L^5N^4(1-LN^2)+D^5L^5N^4+D^4L^4N^3+$$
$$+D^{14}L^7N^6-D^{13}L^6N^4-D^{11}L^6N^3-D^{10}L^5N^2(LN^3-1)+D^9L^5N^4+$$
$$+D^8L^4N^3+D^7L^3N^2-D^5L^3N^2(1-N)-D^4LN\left(L^2N^2+L(1-N)\right)-$$
$$-D^3L^2N^2-D^2L^2N^2-DLN-D^{13}L^6N^4+D^{12}L^5N^3+D^{10}L^5N^2+$$
$$+D^9L^4N(LN^3-1)-D^8L^4N^3-D^7L^3N^2-D^6L^2N+D^4L^2N(1-N)+$$
$$+D^3\left(L^2N^2+L(1-N)\right)+D^2LN+DLN+1=$$
$$=D^{23}L^{13}N^9-2D^{22}L^{12}N^8+D^{21}L^{11}N^7-D^{21}L^{12}N^9+D^{20}L^{11}N^8-D^{20}L^{12}N^7+$$
$$+D^{20}L^{11}N^7(N+1)-D^{19}L^{10}N^6(N+1)-D^{19}L^{11}N^6(LN^3-1)+D^{19}L^{11}N^6-$$

$$-D^{19}L^{10}N^6+D^{18}L^9N^5+D^{18}L^{10}N^5(LN^3-1)+D^{18}L^{11}N^8+D^{18}L^{11}N^7-$$
$$-D^{18}L^9N^5(1-LN^2)+D^{17}L^{10}N^6(LN^3-1)-D^{17}L^{10}N^5(N+1)+$$
$$+D^{17}L^8N^4(1-LN^2)-D^{17}L^{10}N^7+D^{16}L^9N^4-D^{16}L^9N^4(N+1)(LN^3-1)-$$
$$-D^{16}L^{10}N^8+D^{16}L^9N^4+D^{16}L^9N^6-D^{15}L^8N^5+D^{15}L^9N^6(N+1)+$$
$$+D^{15}L^8N^3(LN^3-1)-D^{15}L^9N^7+D^{15}L^8N^3(1-LN^2)-D^{14}L^8N^6+D^{14}L^7N^6-$$
$$-D^{14}L^9N^6(1-N)+D^{14}L^8N^5(N+1)+D^{14}L^7N^2(LN^3-1)(1-LN^2)-$$
$$-D^{14}L^8N^5+D^{14}L^9N^5-D^{13}L^7N^5\left(L^2N^2+L(1-N)\right)-D^{13}L^6N^4-D^{13}L^7N^4+$$
$$+D^{13}L^8N^5(1-N)+D^{13}L^6N^4(LN^3-1)+D^{13}L^7N^4(N+1)-D^{13}L^6N^4-$$
$$-D^{13}L^7N^4(1-LN^2)-D^{12}L^6N^3-2D^{12}L^8N^6+D^{12}L^5N^3-2D^{12}L^8N^6+$$
$$+D^{12}L^6N^4\left(L^2N^2+L(1-N)\right)+D^{12}L^8N^6(1-N)-D^{12}L^6N^3(1-LN^2)+$$
$$+D^{12}L^6N^4\left(L^2N^2+L(1-N)\right)+D^{12}L^5N^3-D^{11}L^8N^6+D^{11}L^7N^5+$$
$$+D^{11}L^6N^5\left(L^2N^2+L(1-N)\right)-D^{11}L^7N^4(1-N^2)-D^{11}L^7N^5-$$
$$-D^{11}L^5N^2(1-LN^2)-D^{11}L^6N^3-D^{10}L^7N^5+D^{10}L^7N^5-D^{10}L^6N^4+$$
$$+D^{10}L^7N^6+D^{10}L^6N^3(1-N)-D^{10}L^5N^3(N+1)\left(L^2N^2+L(1-N)\right)+$$
$$+D^{10}L^5N^2+D^9L^7N^6+D^9L^6N^4+D^9L^7N^6+D^9L^4N^2\left(L^2N^2+L(1-N)\right)+$$
$$+D^9L^5N^4-D^9L^6N^4+D^9L^5N^2(1-N)(1-LN^2)+D^9L^4N(LN^3-1)+$$
$$+D^8L^6N^5+D^8L^3N\left(L^2N^2+L(1-N)\right)(1-LN^2)-D^8L^6N^4+D^8L^5N^3+$$
$$+D^8L^6N^4(1-N)+D^8L^4N^3-D^8L^4N^3+D^7L^4N^2(1-LN^2)+D^7L^3N^2+$$
$$+D^7L^4N^3\left(L^2N^2+L(1-N)\right)-D^7L^3N^2-D^6L^2N+D^6L^4N^2+$$
$$+D^6L^4N^2(1-LN^2)+D^6L^5N^4(1-LN^2)+D^5L^3N(1-LN^2)+$$
$$+D^5L^5N^4-D^5L^3N^2(1-N)+D^4L^4N^3-D^4LN\left(L^2N^2+L(1-N)\right)+$$
$$+D^4L^2N(1-N)-D^3L^2N^2+D^3\left(L^2N^2+L(1-N)\right)+D^2LN-D^2L^2N^2+1=$$
$$=D^{23}L^{13}N^9-2D^{22}L^{12}N^8-D^{21}L^{11}N^7(LN^2-1)+D^{20}L^{11}N^7(2N-L+1)-$$
$$-D^{19}L^{10}N^6(L(LN^3-1)-L+N+2)+D^{18}L^{10}N^5(N^2(1+2LN+L)-1)+$$
$$+D^{17}L^8N^4\left(1-LN\left(N(1-2L)+L(N^2(LN^2-1)-1)\right)\right)+$$
$$+D^{16}L^9N^4\left(3-N(LN^2-N-1)\right)+D^{15}L^8N^3\left(2-N^2(L+1)\right)+$$
$$+D^{14}L^7N^2(N^4(1-2L^2)+LN^2\left(1+N(L+1)\right)-1)-$$
$$-D^{13}L^6N^4(L^2N^2(LN-1)-L(N(N^2+1)-2)+3)-$$

$$-D^{12}L^5N^3\big(L^3N^3(N+4)-2\big(LN\big(L(1-N)\big)+1\big)+L(3-LN^2)\big)-$$
$$-D^{11}L^5N^2(L^3N^4-L^2(N^3+N^5-N^2)-LN(N^2-1)+1)-$$
$$-D^{10}L^5N^2(L^2N^3(N+1)-LN^2(N-2)-1)+$$
$$+D^9L^4N(L^3N^5+L^2N^4+LN(N^2-N+1)-1)-$$
$$-D^8L^4N(LN^2(LN^2-N-1)+N-1)+$$
$$+D^7L^4N^2(L(N(LN^2-N-1)+1)+1)-D^6L^2N\big(1+L^2N(L^2N^4-2)\big)+$$
$$+D^5L^3N\left(L^2N^3+\big(1-N(N(L+1)-1)\big)\right)+D^4L^3N^3(L-1)-D^3L(N-1)+$$
$$+D^2LN(1-LN)+1. \qquad (3.440)$$

Тоді остаточно чисельник співвідношення (3.437), з урахуванням отриманих співвідношень (3.439), (3.440), можна записати наступним чином:

$$X_{nn4}=X_{nn2}+X_{nn3}=2D^{20}L^{11}N^6-2D^{19}L^{10}N^5-6D^{18}L^{10}N^6+$$
$$+2D^{17}L^9N^4(3L^2N-LN+2)+2D^{16}L^8N^3\big(LN\big(1+N^2(1-L)\big)-2\big)+$$
$$+D^{15}L^9N^5(LN+8)-D^{14}L^7N^3(L^2N^2+6LN(1-LN^2)+2N+4)+$$
$$+D^{13}L^7N^2\big(4(1-LN^2)-LN^3(3LN+4)\big)+$$
$$+D^{12}L^7N^3\big(LN^2(2-N)+2(LN+2)\big)+$$
$$+D^{11}L^6N^3\big(L(N^2(LN-1)+N-2)+2(1-LN^2)\big)+$$
$$+D^{10}L^6N^3(-3LN^2-N+2)+D^9L^5N^2(LN(N^2-4N)-LN+2)+$$
$$+D^8L^5N^2\big(N^2\big(1-L(N+3)\big)+2\big)+D^7L^5N^2(N^2(3-LN)-3N+2)+$$
$$+2D^6L^4N+D^4L^3N^2+D^{23}L^{13}N^9-2D^{22}L^{12}N^8-D^{21}L^{11}N^7(LN^2-1)+$$
$$+D^{20}L^{11}N^7(2N-L+1)-D^{19}L^{10}N^6(L(LN^3-1)-L+N+2)+$$
$$+D^{18}L^{10}N^5(N^2(1+2LN+L)-1)+$$
$$+D^{17}L^8N^4\left(1-LN\big(N(1-2L)+L(N^2(LN^2-1)-1)\big)\right)+$$
$$+D^{16}L^9N^4\big(3-N(LN^2-N-1)\big)+D^{15}L^8N^3\big(2-N^2(L+1)\big)+$$
$$+D^{14}L^7N^2(N^4(1-2L^2)+LN^2\big(1+N(L+1)\big)-1)-$$
$$-D^{13}L^6N^4(L^2N^2(LN-1)-L(N(N^2+1)-2)+3)-$$
$$-D^{12}L^5N^3\big(L^3N^3(N+4)-2\big(LN\big(L(1-N)\big)+1\big)+L(3-LN^2)\big)\big)-$$
$$-D^{11}L^5N^2(L^3N^4-L^2(N^3+N^5-N^2)-LN(N^2-1)+1)-$$

$$-D^{10}L^5N^2(L^2N^3(N+1)-LN^2(N-2)-1)+$$
$$+D^9L^4N(L^3N^5+L^2N^4+LN(N^2-N+1)-1)-$$
$$-D^8L^4N(LN^2(LN^2-N-1)+N-1)+$$
$$+D^7L^4N^2(L(N(LN^2-N-1)+1)+1)-D^6L^2N\big(1+L^2N(L^2N^4-2)\big)+$$
$$+D^5L^3N\Big(L^2N^3+\big(1-N(N(L+1)-1)\big)\Big)+D^4L^3N^3(L-1)-D^3L(N-1)+$$
$$+D^2LN(1-LN)+1=D^{23}L^{13}N^9-2D^{22}L^{12}N^8-D^{21}L^{11}N^7(LN^2-1)+$$
$$+D^{20}L^{11}N^6(N(2N-L+1)+2)-$$
$$-D^{19}L^{10}N^5\big(2+N(L(LN^3-1)-L+N+2)\big)+$$
$$+D^{18}L^{10}N^5\big((N^2(1+2LN+L)-1)-6N\big)+$$
$$+D^{17}L^8N^4\big(1-LN\big(N(1-2L)+L(N^2(LN^2-1)-1)\big)+$$
$$+2L(3L^2N-LN+2)\big)+D^{16}L^8N^3\big(LN\big(3-N(LN^2-N-1)\big)+$$
$$+2\big(LN\big(1+N^2(1-L)\big)-2\big)\Big)+D^{15}L^8N^3\Big(\big(2-N^2(L+1)\big)+LN^2(LN+8)\Big)+$$
$$+D^{14}L^7N^2\Big((N^4(1-2L^2)+LN^2\big(1+N(L+1)\big)-1)-N\,(L^2N^2+$$
$$+6LN(1-LN^2)+2N+4)\big)+D^{13}L^6N^2\big(L\big(4(1-LN^2)-LN^3(3LN+4)\big)-$$
$$-N^2(L^2N^2(LN-1)-L(N(N^2+1)-2)+3)\big)-D^{12}L^5N^3(L^3N^3(N+4)-$$
$$-2\big(LN\big(L(1-N)\big)+1\big)+L(3-LN^2)+L^2\big(LN^2(2-N)+2(LN+2)\big)\Big)+$$
$$+D^{11}L^5N^2(LN(L(N^2(LN-1)+N-2)+2(1-LN^2)-L^3N^4-$$
$$-L^2(N^3+N^5-N^2)-LN(N^2-1)+1)\big)+D^{10}L^5N^2\big(-L^2N^3\big(3+(N+1)\big)+$$
$$+LN^2(N-3)+2LN+1)+D^9L^5N^2(LN^3(LN+1)-LN(N-5)+$$
$$+N(N-1)-1)+D^8L^4N(-L^2N^3(2N+3)+LN(2N^2+N+2)-N+1)+$$
$$+D^7L^4N^2(2LN(N-4)+3L+1)-D^6L^2N(L^2N(L^2N-3)+1)+$$
$$+D^5L^3N\Big(L^2N^3+\big(1-N(N(L+1)-1)\big)\Big)+D^4L^3N^2(N(L-1)+1)-$$
$$-D^3L(N-1)+D^2LN(1-LN)+1. \qquad (3.441)$$

Проаналізуємо тепер знаменник співвідношення (3.437). В результаті множення поліномів та зведення подібних доданків, з урахуванням співвідношень (3.427), (3.438), отримуємо наступний аналітичний вираз у поліноміальній формі:

$$X_{nd3} = (1 - D^3LN)X_{cnhd}X_{nhd12} = (1 - D^3LN)(-D^{10}L^7N^5 + D^9L^6N^4 +$$
$$+D^8L^6N^5 - D^7L^5N^3(1 + N) + D^6L^4N^2 + D^5L^3N(1 - LN^2) + D^4L^4N^3 - DLN +$$
$$+1)(D^9L^5N^3 - 3D^7L^4N^3 + 2D^6L^3N + D^5L^3N^3 + D^4L^2N^2) =$$
$$= (1 - D^3LN)(-D^{19}L^{12}N^8 + D^{18}L^{11}N^7 + D^{17}L^{11}N^8 - D^{16}L^{10}N^6(1 + N) +$$
$$+D^{15}L^9N^5 + D^{14}L^8N^4(1 - LN^2) + D^{13}L^9N^6 - D^{10}L^6N^4 + D^9L^5N^3 +$$
$$+3D^{17}L^{11}N^8 - 3D^{16}L^{10}N^7 - 3D^{15}L^{10}N^8 + 3D^{14}L^9N^6(1 + N) - 3D^{13}L^8N^5 -$$
$$-3D^{12}L^7N^4(1 - LN^2) - 3D^{11}L^8N^6 + 3D^8L^5N^4 - 3D^7L^4N^3 - 2D^{16}L^{10}N^6 +$$
$$+2D^{15}L^9N^5 + 2D^{14}L^9N^6 - 2D^{13}L^8N^4(1 + N) + 2D^{12}L^7N^3 +$$
$$+2D^{11}L^6N^2(1 - LN^2) + 2D^{10}L^7N^4 - 2D^7L^4N^2 + 2D^6L^3N -$$
$$-D^{15}L^{10}N^8 + D^{14}L^9N^7 + D^{13}L^9N^8 - D^{12}L^8N^6(1 + N) + D^{11}L^7N^5 +$$
$$+D^{10}L^6N^4(1 - LN^2) + D^9L^7N^6 - D^6L^4N^4 + D^5L^3N^3 - D^{14}L^9N^7 +$$
$$+D^{13}L^8N^6 + D^{12}L^8N^7 - D^{11}L^7N^5(1 + N) + D^{10}L^6N^4 +$$
$$+D^9L^5N^3(1 - LN^2) + D^8L^6N^5 - D^5L^3N^3 + D^4L^2N^2) =$$
$$= -D^{19}L^{12}N^8 + D^{18}L^{11}N^7 + D^{17}L^{11}N^8 - D^{16}L^{10}N^6(1 + N) +$$
$$+D^{15}L^9N^5 + D^{14}L^8N^4(1 - LN^2) + D^{13}L^9N^6 - D^{10}L^6N^4 + D^9L^5N^3 +$$
$$+3D^{17}L^{11}N^8 - 3D^{16}L^{10}N^7 - 3D^{15}L^{10}N^8 + 3D^{14}L^9N^6(1 + N) - 3D^{13}L^8N^5 -$$
$$-3D^{12}L^7N^4(1 - LN^2) - 3D^{11}L^8N^6 + 3D^8L^5N^4 - 3D^7L^4N^3 - 2D^{16}L^{10}N^6 +$$
$$+2D^{15}L^9N^5 + 2D^{14}L^9N^6 - 2D^{13}L^8N^4(1 + N) + 2D^{12}L^7N^3 +$$
$$+2D^{11}L^6N^2(1 - LN^2) + 2D^{10}L^7N^4 - 2D^7L^4N^2 + 2D^6L^3N -$$
$$-D^{15}L^{10}N^8 + D^{14}L^9N^7 + D^{13}L^9N^8 - D^{12}L^8N^6(1 + N) + D^{11}L^7N^5 +$$
$$+D^{10}L^6N^4(1 - LN^2) + D^9L^7N^6 - D^6L^4N^4 + D^5L^3N^3 - D^{14}L^9N^7 +$$
$$+D^{13}L^8N^6 + D^{12}L^8N^7 - D^{11}L^7N^5(1 + N) + D^{10}L^6N^4 +$$
$$+D^9L^5N^3(1 - LN^2) + D^8L^6N^5 - D^5L^3N^3 + D^4L^2N^2 +$$
$$+D^{22}L^{13}N^9 - D^{21}L^{12}N^8 - D^{20}L^{12}N^9 + D^{19}L^{11}N^7(1 + N) -$$
$$-D^{18}L^{10}N^6 - D^{17}L^9N^5(1 - LN^2) - D^{16}L^{10}N^7 + D^{13}L^7N^5 - D^{12}L^6N^4 -$$
$$-3D^{20}L^{12}N^9 + 3D^{19}L^{11}N^8 + 3D^{18}L^{11}N^9 - 3D^{17}L^{10}N^7(1 + N) + 3D^{16}L^9N^6 +$$
$$+3D^{15}L^8N^5(1 - LN^2) + 3D^{14}L^9N^7 - 3D^{11}L^6N^5 + 3D^{10}L^5N^4 + 2D^{19}L^{11}N^7 -$$
$$-2D^{18}L^{10}N^6 - 2D^{17}L^{10}N^7 + 2D^{16}L^9N^5(1 + N) - 2D^{15}L^8N^4 -$$
$$-2D^{14}L^7N^3(1 - LN^2) - 2D^{13}L^8N^5 + 2D^{10}L^5N^3 - 2D^9L^4N^2 +$$

$$+D^{18}L^{11}N^9 - D^{17}L^{10}N^8 - D^{16}L^{10}N^9 + D^{15}L^9N^7(1+N) - D^{14}L^8N^6 -$$
$$- D^{13}L^7N^5(1-LN^2) - D^{12}L^8N^7 + D^9L^5N^5 - D^8L^4N^4 + D^{17}L^{10}N^8 -$$
$$- D^{16}L^9N^7 - D^{15}L^9N^8 + D^{14}L^8N^6(1+N) - D^{13}L^7N^5 -$$
$$- D^{12}L^6N^4(1-LN^2) - D^{11}L^7N^6 + D^8L^4N^4 - D^7L^3N^3 =$$
$$= D^{22}L^{13}N^9 - D^{21}L^{12}N^8 - D^{20}L^{12}N^9 - 3D^{20}L^{12}N^9 + D^{19}L^{11}N^7(1+N) +$$
$$+ 3D^{19}L^{11}N^8 - D^{19}L^{12}N^8 + 2D^{19}L^{11}N^7 + D^{18}L^{11}N^7 - D^{18}L^{10}N^6 +$$
$$+ 3D^{18}L^{11}N^9 - 2D^{18}L^{10}N^6 + D^{18}L^{11}N^9 + D^{17}L^{11}N^8 - D^{17}L^9N^5(1-LN^2) -$$
$$- 3D^{17}L^{10}N^7(1+N) - 2D^{17}L^{10}N^7 - D^{17}L^{10}N^8 + D^{17}L^{10}N^8 -$$
$$- D^{16}L^{10}N^6(1+N) - 3D^{16}L^{10}N^7 - 2D^{16}L^{10}N^6 - D^{16}L^{10}N^7 + 3D^{16}L^9N^6 +$$
$$+ 2D^{16}L^9N^5(1+N) - D^{16}L^{10}N^9 - D^{16}L^9N^7 + D^{15}L^9N^5 + 2D^{15}L^9N^5 -$$
$$- 3D^{15}L^{10}N^8 - D^{15}L^{10}N^8 + 3D^{14}L^9N^6(1+N) + D^{14}L^8N^4(1-LN^2) +$$
$$+ 2D^{14}L^9N^6 + D^{14}L^9N^7 - D^{14}L^9N^7 + 3D^{14}L^9N^7 - 2D^{14}L^7N^3(1-LN^2) +$$
$$+ D^{14}L^8N^6(1+N) - D^{14}L^8N^6 - 2D^{13}L^8N^4(1+N) - 3D^{13}L^8N^5 +$$
$$+ D^{13}L^9N^6 + D^{13}L^9N^8 + D^{13}L^8N^6 + D^{13}L^7N^5 - 2D^{13}L^8N^5 -$$
$$- D^{13}L^7N^5(1-LN^2) - D^{13}L^7N^5 + 2D^{12}L^7N^3 - D^{12}L^8N^7 -$$
$$- 3D^{12}L^7N^4(1-LN^2) - D^{12}L^8N^6(1+N) + D^{12}L^8N^7 - D^{12}L^6N^4 -$$
$$- D^{12}L^6N^4(1-LN^2) - 3D^{11}L^8N^6 + 2D^{11}L^6N^2(1-LN^2) + D^{11}L^7N^5$$
$$- D^{11}L^7N^5(1+N) - 3D^{11}L^6N^5 - D^{11}L^7N^6 + 2D^{10}L^7N^4 - D^{10}L^6N^4 +$$
$$+ D^{10}L^6N^4(1-LN^2) + D^{10}L^6N^4 + 3D^{10}L^5N^4 + 2D^{10}L^5N^3 + D^9L^5N^5 +$$
$$+ D^9L^7N^6 + D^9L^5N^3 + D^9L^5N^3(1-LN^2) + D^9L^5N^3(1-LN^2) - 2D^9L^4N^2 +$$
$$+ D^8L^6N^5 + 3D^8L^5N^4 + D^8L^6N^5 - D^8L^4N^4 + D^8L^4N^4 - 2D^7L^4N^2 - 3D^7L^4N^3 -$$
$$- D^7L^3N^3 - D^6L^4N^4 + 2D^6L^3N + D^5L^3N^3 - D^5L^3N^3 + D^4L^2N^2 =$$
$$= D^{22}L^{13}N^9 - D^{21}L^{12}N^8 - 4D^{20}L^{12}N^9 + D^{19}L^{11}N^7\big(3 + N(4-L)\big) +$$
$$+ D^{18}L^{10}N^6(LN(1+4N^2) - 3) + D^{17}L^9N^5(LN^2(N(3-L)+2) - 1) +$$
$$+ D^{16}L^9N^5(N(L(N(N^2-1)-3) - N + 5) + 2) + D^{15}L^9N^5(3 - 4LN^3) +$$
$$+ D^{14}L^7N^3\Big(N\big(L^2N^2(2N+1) + L(N(N^2+2)+1)\big) - 2\Big) +$$
$$+ D^{13}L^7N^5(LN(LN^5(N^2+1) + N(N-7) - 2) - 1 + LN^2) +$$
$$+ D^{12}L^6N^3\big(LN^3\big(3 - L(N+1)\big) + N(N-5) + 2\big) -$$
$$- D^{11}L^6N^2(LN^2(N^2(3L+1)+1) + 3N - 2) +$$

$$+D^{10}L^5N^3(LN(L(-N^2+2)+1)+3N+2)+$$
$$+D^9L^4N^2\left(LN\left(L^2N^2-\left((2L-1)N^2-3\right)\right)-2\right)+D^8L^5N^4(2LN+3)-$$
$$-D^7L^3N^2(L(N+1)+N)-D^6L^3N(LN^3-2)+D^4L^2N^2. \quad (3.442)$$

Тоді, згідно із співвідношеннями (3.435), з урахуванням отриманих поліноміальних функцій (3.436) – (3.442), які залежать лише від параметрів згорткового коду D, L та N, для залежності між станами автомату $X_n(X_h)$ можна записати наступний аналітичний вираз у вигляді дрібно-раціональної функції, явно вираженої відносно параметрів коду:

$$X_n = \frac{X_{nn}\ X_{nd}}{X_{nd}\ X_{nd3}} X_h = T_{nh}(D, L, N)X_h, \quad (3.443)$$

де

$$T_{nh}(D, L, N) = \frac{X_{nn}\ X_{nd}}{X_{nd2}X_{nd3}}. \quad (3.444)$$

Функція $T_{nh}(D, L, N)$ у співвідношеннях (3.443), (3.444) є передавальною функцією між станами скінченного автомату X_h та X_n.

Зрозуміло, що поліноміальні множники у чисельнику та знаменнику дрібно-раціональної передавальної функції $T_{nh}(D, L, N)$ також мають бути обчислені, але одна з загальних проблем проведення комбінаторних обчислень в дискретній математиці полягаю в тому, що множення поліномів високих порядків є дуже кропіткою та громіздкою роботою, в процесі якої, у разі її виконання вручну, можуть виникати обчислювальні помилки. Тому зазвичай для проведення алгебраїчних операцій дискретної математики з поліномами високих порядків застосовують ефективні та надійні комп'ютерні обчислювальні системи. Найбільш ефективним для здійснення алгебраїчних перетворень з поліноміальними функціями є використання аналітичних процесорів сучасних комп'ютерних систем, призначених для виконання інженерних та науково-технічних розрахунків. Серед таких програмних продуктів найбільш популярними серед науковців є система аналітичних розрахунків Maple та символьний процесор системи науково-технічних

розрахунків MatLab [13, 14]. Тому проводити подальші алгебраїчні операції з поліномами високих порядків для прикладу, який розглядається, будемо з використанням програмних засобів для роботи з поліномами, реалізованих в символьному процесорі системи науково-технічних розрахунків MatLab [13, 14]. Результат множення поліномів, записаних в чисельнику та знаменнику співвідношення (3.444), наведений у додатку Р. Результатом множення поліномів у чисельнику співвідношення (3.444) є символьна змінна **XNNOM**, а результатом множення поліномів у знаменнику цього співвідношення – символьна змінна **XNDEN**. Зрозуміло, що змінна **XNNOM** відповідає поліноміальній функції сорок другого порядку відносно змінної *D*, а змінна **XNDEN** – поліноміальній функції сорок першого порядку. З урахуванням результатів проведених комп'ютерних розрахунків отримане аналітичне співвідношення (3.444) для функціональної залежності між станами скінченного автомату $T_{nh}(D, L, N)$ можна переписати наступним чином:

$$T_{nh}(D, L, N) = \frac{X_{nn}\ X_{nd}}{X_{nd}\ X_{nd3}} = \frac{X_{nnom}(D,L,N)}{X_{nden}(D,L,N)}, \qquad (3.445)$$

де $X_{nnom}(D,\ L,\ N)$ – поліном, що відповідає символьній змінній **XNNOM**, а – $X_{nden}(D, L, N)$ – поліном, що відповідає символьній змінній **XNDEN**. Оскільки поліноміальні функції високих порядків є досить громіздкими, надалі під час виконання подібних операцій над поліномами не будемо виводити на екран результат роботи символьного процесора, а виведемо та проаналізуємо лише остаточний результат, який відповідає передавальній функції скінченного автомату (3.404). Для запису математичних виразів у поліноміальній формі використовується команда символьного процесору **expand(Ім'я_змінної)**, а для зведення подібних доданків та формування поліноміальних коефіцієнтів за степенями змінної **D** за їхнім зростанням – команда символьного процесору **collect(Ім'я_змінної, D)** [13, 14]. Параметр цих команд **Ім'я_змінної** відповідає символьній змінній, яка описує відповідний поліном.

Щодо поліноміальних операцій додавання, множення та ділення поліномів, а також піднесення до степені, в системі науково-технічних розрахунків MatLab вони здійснюються звичайним чином з використанням символів +, –, *, / та ^ [13, 14]. Наприклад, якщо необхідно помножити два поліноми,

$$P1(D,L,N) = 2D^5L^4N^5(LN+1) - 5D^4L^3N + 5$$

та

$$P2(D,L,N) = 3D^4L^4N^5(L^2N+1) - 2D^3L^3N - 3,$$

відповідні командні рядки символьного процесору системи MatLab матимуть наступний вигляд:

```
>> syms D L N;
>> P1=2*D^5*L^4*N^5*(L*N+1)-5*D^4*L^3*N+5;
>> P2=3*D^4*L^4*N^5*(L^2*N+1)-2*D^3*L^3*N-3;
>> P=P1*P2
P =
-(2*D^5*L^4*N^5*(L*N + 1) - 5*D^4*L^3*N +
 5)*(2*D^3*L^3*N - 3*D^4*L^4*N^5*(N*L^2 + 1) + 3)
>> PE=expand(P)
PE =
6*D^9*L^11*N^12 + 6*D^9*L^10*N^11 + 6*D^9*L^9*N^11 +
6*D^9*L^8*N^10 - 15*D^8*L^9*N^7 - 4*D^8*L^8*N^7 -
19*D^8*L^7*N^6 + 10*D^7*L^6*N^2 - 6*D^5*L^5*N^6 -
6*D^5*L^4*N^5 + 15*D^4*L^6*N^6 + 15*D^4*L^4*N^5 +
15*D^4*L^3*N - 10*D^3*L^3*N - 15
>> PC=collect(PE, D)
PC =
(6*L^11*N^12 + 6*L^10*N^11 + 6*L^9*N^11 +
6*L^8*N^10)*D^9 + (-15*L^9*N^7 - 4*L^8*N^7 -
19*L^7*N^6)*D^8 + 10*L^6*N^2*D^7 + (-6*L^5*N^6 -
```

```
6*L^4*N^5)*D^5 + (15*L^6*N^6 + 15*L^4*N^5 +
15*L^3*N)*D^4 + (-10*L^3*N)*D^3 - 15
```

За умови відомої функціональної залежності між станами скінченного автомату $T_{nh}(D, L, N)$, заданої співвідношеннями (3.443) – (3.445), з урахуванням отриманих залежностей між станами автомату (3.405), (3.406), (3.412), (3.413), (3.414), (3.417), (3.431), а також початкової системи лінійних рівнянь (3.403), можна знайти залежності між іншими станами автомату та станом X_h лише через параметри згорткового коду D, L та N. Спочатку, з використанням співвідношень (3.427) – (3.431), (3.445), визначимо залежність між станами скінченного автомату $X_c(X_h)$. Відповідно, маємо наступний результат:

$$X_c(X_h) = \frac{X_{cnn}}{X_{cnhd}} X_n + \frac{X_{chn}}{(1 - D^3LN)X_{cnhd}} X_h =$$

$$= \left(\frac{X_{nnom}X_{cnn}}{X_{nden}X_{cnhd}} + \frac{X_{chn}}{(1 - D^3LN)X_{cnhd}}\right) X_h =$$

$$= \frac{(1-D^3LN)X_{nnom}X_{cnn}+X_{chn}X_{nden}}{(1-D^3LN)X_{nden}X_{cnhd}} X_h = \frac{X_{cnom}}{X_{cden}} X_h. \quad (3.446)$$

Відповідні командні рядки системи науково-технічних розрахунків MatLab, в яких проведені обчислення поліноміальних змінних $X_{cnom}(D, L, N)$ та $X_{cnom}(D, L, N)$ згідно із співвідношенням (3.446), наведені у додатку Р. Оскільки ці результати розрахунків є проміжними, вони не виводяться на екран і не роздруковуються.

Для обчислення залежності між станами скінченного автомату $X_b(X_h)$ скористаємося співвідношенням (3.424). Для цього введемо наступні додаткові змінні:

$$X_{bc}(D, L, N) = -D^9L^6N^4 + D^7L^5N^4 - D^3L^3N^2,$$

$$X_{bnnom}(D, L, N) = -D^{11}L^7N^4 + D^9L^6N^4(1 + N^2) - D^8L^5N^3 -$$

$$-D^7L^5N^4 - D^5L^4N^2 + D^3L^3N^2 - D^2L^2N, \quad (3.447)$$

$$X_{bhnom}(D, L, N) = D^{10}L^5N^2 - D^7L^4N - D^6L^4N^2 + D^5L^4N^2 - D^2L^2N.$$

З урахуванням нових поліноміальних змінних, заданих співвідношеннями

(3.447), а також отриманих залежностей $X_n(X_h)$ та $X_c(X_h)$, заданих співвідношеннями (3.443) – (3.446), перепишемо співвідношення (3.424) наступним чином:

$$D^2LX_b(D,L,N,X_h) = X_{bc}X_c + \frac{X_{bnnom}}{1-DLN}X_n - \frac{X_{bhnom}}{1-D^3LN}X_h =$$

$$= \frac{X_{bc}X_{cnom}}{X_{cden}}X_h + \frac{X_{bnnom}X_{nnom}}{(1-DLN)X_{nden}}X_h - \frac{X_{bhnom}}{1-D^3LN}X_h,$$

або

$$X_b(D,L,N,X_h) = \left(\frac{X_{bc}X_{cnom}}{D^2LX_{cden}} + \frac{X_{bnnom}X_{nnom}}{(1-DLN)D^2LX_{nden}} - \frac{X_{bhnom}}{D^2L(1-D^3LN)}\right)X_h =$$

$$= \frac{X_{bc}X_{cnom}X_{nden}(1-DLN)(1-D^3LN)}{D^2L(1-D^3LN)(1-DLN)X_{nden}X_{cden}}X_h +$$

$$+ \frac{(1-D^3LN)X_{cden}X_{bnnom}X_{nnom}}{D^2L(1-D^3LN)(1-DLN)X_{nden}X_{cden}}X_h +$$

$$+ \frac{(DLN-1)X_{nden}X_{cden}X_{bhnom}}{D^2L(1-D^3LN)(1-DLN)X_{nden}X_{cden}}X_h. \qquad (3.448)$$

Як і раніше, для спрощення подальших аналітичних перетворень з поліномами, запишемо чисельник та знаменник функціональної залежності між станами скінченного автомату $X_b(X_h)$, заданої співвідношенням (3.448), як окремі поліноміальні функції від параметрів згорткового коду D, L та N. Відповідно, маємо:

$$X_{bnom}(D,L,N) = X_{bc}X_{cnom}X_{nden}(1-DLN)(1-D^3LN) +$$

$$+(1-D^3LN)X_{cden}X_{bnnom}X_{nnom} + (DLN-1)X_{nden}X_{cden}X_{bhnom},$$

$$X_{bden}(D,L,N) = D^2L(1-D^3LN)(1-DLN)X_{nden}X_{cden}, \qquad (3.449)$$

$$X_b(D,L,N,X_h) = \left(\frac{X_{bnom}(D,L,N)}{X_{bden}(D,L,N)}\right)X_h.$$

Обчислення поліноміальних коефіцієнтів для функціональної залежності між станами скінченного автомату $X_b(X_h)$, згідно із співвідношеннями (3.447), (3.449), також проведені у додатку Р.

За умови відомих залежностей між станами скінченного автомату $X_n(X_h)$ та

$X_c(X_h)$, заданих співвідношеннями (3.443) – (3.446), залежність $X_l(X_h)$ можна знайти через аналіз співвідношення (3.417) та алгебраїчні перетворення поліноміальних виразів. Відповідно, маємо:

$$X_l = \frac{D^3L^2N^2X_c + (D^5L^3N^2 - D^3L^2N^2 + D^2LN)X_n}{1 - DLN} =$$

$$= \frac{D^3L^2N^2\frac{X_{cnom}}{X_{cden}} + (D^5L^3N^2 - D^3L^2N^2 + D^2LN)\frac{X_{nnom}}{X_{nden}}}{1 - DLN} =$$

$$= \left(\frac{D^3L^2N^2X_{cnom}}{(1 - DLN)X_{cden}} + \frac{(D^5L^3N^2 - D^3L^2N^2 + D^2LN)X_{nnom}}{(1 - DLN)X_{nden}}\right)X_h =$$

$$= \left(\frac{D^3L^2N^2X_{cnom}X_{nden}}{(1 - DLN)X_{cden}X_{nden}} + \frac{(D^5L^3N^2 - D^3L^2N^2 + D^2LN)X_{cden}X_{nnom}}{(1 - DLN)X_{cden}X_{nden}}\right)X_h =$$

$$= \frac{D^3L^2N^2X_{cnom}X_{nden} + (D^5L^3N^2 - D^3L^2N^2 + D^2LN)X_{cden}X_{nnom}}{(1 - DLN)X_{cden}X_{nden}}X_h =$$

$$= \frac{X_{lnom}}{X_{lden}}X_h. \qquad (3.450)$$

Обчислення поліноміальних коефіцієнтів для поліномів $X_{lnom}(D, L, N)$ та $X_{ldem}(D, L, N)$ згідно з співвідношенням (3.450) також проводилось у додатку Р, але результати цих проміжних обчислень не виводились на екран.

За умови відомої залежності між станами скінченного автомату $X_l(X_h)$, заданої співвідношенням (3.450), залежність $X_g(X_h)$ можна знайти зі співвідношення (3.412), залежність $X_d(X_h)$ – зі співвідношення (3.413), а залежність $X_f(X_h)$ – відповідно, зі співвідношення (3.414). Прості алгебраїчні перетворення дають наступні результати:

$$X_d = \frac{X_h - D^2LNX_l}{D^2LN} = \frac{1 - D^2LN\frac{X_{lnom}}{X_{lden}}}{D^2LN}X_h = \frac{X_{lden} - D^2LNX_{lnom}}{D^2LN_{\ lden}}X_h = \frac{X_{dnom}}{X_{dden}}X_h.$$

$$X_g = \frac{X_h + (D^4L^2N - D^2LN)X_l}{D^2LN} = \frac{1 + (D^4L^2N - D^2LN)\frac{X_{lnom}}{X_{lden}}}{D^2LN}X_h =$$

$$= \frac{X_{lden} + (D^4L^2N - D^2LN)X_{lnom}}{D^2LNX_{lden}} = \frac{X_{gnom}}{X_{gden}}X_h.$$

$$X_f = \frac{X_l - D^2LN_{\ n}}{D^2LN} = \left(\frac{X_{lnom}}{D^2LNX_{lden}} - \frac{D^2LN_{\ nnom}}{D^2LNX_{nden}}\right)X_h =$$

$$= \left(\frac{X_{lnom}X_{nden}}{D^2LNX_{lden}X_{nden}} - \frac{D^2LN_{\ nnom}X_{lden}}{D^2LNX_{lden}X_{nden}}\right)X_h =$$

$$= \frac{X_{lnom}X_{nden} - D^2LN_{\ nnom}X_{lden}}{D^2LNX_{lden}X_{nden}}X_h = \frac{X_{fnom}}{X_{fden}}X_{h.} \qquad (3.451)$$

Розрахунок значень поліноміальних коефіцієнтів для функцій $X_{dnom}(D, L, N)$, $X_{ddem}(D, L, N)$, $X_{gnom}(D, L, N)$, $X_{gdem}(D, L, N)$, $X_{fnom}(D, L, N)$, $X_{fdem}(D, L, N)$, заданих співвідношеннями (3.451), також був проведений з використанням засобів символьного процесора системи науково-технічних розрахунків MatLab, відповідні командні рядки наведені у додатку Р.

За умови відомих знайдених функцій для шести станів скінченного автомату $X_n(X_h)$, $X_c(X_h)$, $X_l(X_h)$, $X_g(X_h)$, $X_d(X_h)$ та $X_f(X_h)$, можна легко виразити через поліноми та дрібно-раціональні функції і залежності для всіх інших станів автомату, використовуючи для проведення аналітичних перетворень базову систему рівнянь (4.403). Наприклад, для функції $X_k(X_h)$, з урахуванням співвідношень (3.443) – (3.446), (3.451) можна записати:

$$X_k = X_f + D^2LX_n = \left(\frac{X_{fnom}}{X_{fden}} + \frac{D^2LX_{nnom}}{X_{nden}}\right)X_h =$$

$$\frac{X_{fnom}X_{nden} + D^2LX_{nnom}}{X_{fden}X_{nden}}X_h = \frac{X_{knom}}{X_{kden}}X_h, \qquad (3.452)$$

тоді, відповідно, для залежності $X_e(X_h)$ маємо наступний результат:

$$X_e = D^3LX_c + X_k = \left(\frac{D^3LX_{cnom}}{X_{cden}} + \frac{X_{knom}}{X_{kden}}\right)X_h =$$

$$= \frac{D^3LX_{kden}X_{cnom} + X_{knom}X_{cden}}{X_{cden}X_{kden}}X_h = \frac{X_{enom}}{X_{eden}}X_h. \qquad (3.453)$$

Для пошуку аналітичної залежності $X_m(X_h)$ у формі дрібно-раціональної функції скористаємося отриманими співвідношеннями (3.451) та (3.406). Відповідно, маємо:

$$X_m = X_g + D^2LX_o = \frac{X_{gnom}}{X_{gden}}X_h + \frac{-D^6L^2N + D^3L + D^2LN}{1 - D^3LN}X_h =$$

$$= \frac{X_{gnom}(1 - D^3LN)}{X_{gden}(1 - D^3LN)}X_h + \frac{(-D^6L^2N + D^3L + D^2LN)X_{gden}}{(1 - D^3LN)X_{gden}}X_h =$$

$$= \frac{(1-D^3LN)X_{gnom}+(-D^6L^2N+D^3L+D^2LN)X_{gden}}{(1-D^3LN)X_{gden}}X_h = \frac{X_{mnom}}{X_{mden}}X_h. \quad (3.454)$$

Враховуючи аналітичні співвідношення (3.453) та (3.454), отримані для пошуку залежності між станами скінченного автомату $X_i(X_h)$, з урахуванням восьмого рівняння системи (4.403), можна записати наступний аналітичний вираз:

$$X_i = D^3LX_e + DL\ _m = D^3L\frac{X_{enom}}{X_{eden}}X_h + DL\frac{X_{mnom}}{X_{mden}}X_h =$$

$$= \left(\frac{D^3LX_{enom}}{X_{eden}} + \frac{DLX_{mnom}}{X_{mden}}\right)X_h =$$

$$= \frac{D^3LX_{enom}X_{mden} + DLX_{mnom}X_{eden}}{X_{eden}X_{mden}}X_h =$$

$$= \frac{DL(D^2X_{enom}X_{mden}+X_{mnom}X_{eden})}{X_{eden}X_{mden}} = \frac{X_{inom}}{X_{iden}}X_h. \quad (3.455)$$

Розрахунок значень поліноміальних коефіцієнтів за наведеними співвідношеннями (3.452) – (3.455) також реалізований в системі науково-технічних розрахунків MatLab, відповідні командні рядки наведені у додатку Р.

З урахуванням отриманих співвідношень для станів скінченного автомату (3.449) та (3.455), перепишемо передавальну функцію скінченного автомату, задану аналітичним співвідношенням (3.404), наступним чином:

$$T(D,L,N) = \frac{(D^2LN)D^2LX_i}{X_b - D^2LNX_i} = \frac{D^4L^2N\frac{X_{inom}}{X_{iden}}X_h}{\frac{X_{bnom}}{X_{bden}}X_h - D^2LN\frac{X_{inom}}{X_{iden}}X_h} = \frac{D^4L^2N\frac{X_{inom}}{X_{iden}}}{\frac{X_{bnom}}{X_{bden}} - D^2LN\frac{X_{inom}}{X_{iden}}} =$$

$$= \frac{D^4L^2N\frac{X_{inom}}{X_{iden}}}{\frac{X_{bnom}X_{iden}}{X_{bden}X_{iden}} - \frac{D^2LNX_{inom}X_{bden}}{X_{iden}X_{bden}}} = \frac{D^4L^2N\frac{X_{inom}}{X_{iden}}}{\frac{X_{bnom}X_{iden} - D^2LNX_{inom}X_{bden}}{X_{iden}X_{bden}}} =$$

$$= \frac{D^4 L^2 N X_{inom} X_{bden}}{X_{bnom} X_{iden} - D^2 L N X_{inom} X_{bden}} = \frac{T_{nom}(D,L,N)}{T_{den}(D,L,N)}. \quad (3.456)$$

Розрахунок значень поліноміальних коефіцієнтів для функцій $T_{nom}(D, L, N)$ та $T_{den}(D, L, N)$, заданих співвідношенням (3.456), також був проведений з використанням символьного процесору системи науково-технічних розрахунків MatLab, відповідні командні рядки наведені у додатку Р. Оскільки ці поліноми є остаточним результатом виконаних розрахунків, відповідні символьні змінні були виведені до файлу та роздруковані. Виведення поліномів високих порядків на екран в системі науково-технічних розрахунків MatLab є неможливим із-за існуючих обмежень на довжину рядка, який виводиться. Максимальна довжина такого рядка складає 1024 символи [13, 14]. Для роботи з файлами з метою запису результатів аналітичних розрахунків коефіцієнтів поліномів використовувались стандартні функції системи науково-технічних розрахунків MatLab **`fopen()`**, **`fwrite()`** та **`fclose()`** [13, 14]. Отримані результати розрахунків поліноміальних коефіцієнтів для поліномів $T_{nom}(D, L, N)$ та $T_{den}(D, L, N)$ також наведені у додатку Р. Особливості проведення розрахунків за співвідношеннями (3.446) – (3.456) з використанням символьного процесора системи науково-технічних розрахунків MatLab будуть описані у підрозділі 3.8.13.5.

Як видно з результатів розрахунків, наведених у додатку Р, поліноми $T_{nom}(D, L, N)$ та $T_{den}(D, L, N)$, які являють собою чисельник та знаменник передавальної функції скінченного автомату для згорткового коду з параметрами $K = 5$, $\frac{1}{n} = \frac{1}{3}$, структурна схема якого наведена на рис. 3.108, записуються наступним чином:

$$T_{nom}(D, L, N) = (L^{269}N^{183} - L^{271}N^{186})D^{448} + (25L^{270}N^{185} -$$
$$-L^{270}N^{184} + 8L^{269}N^{185} - 25L^{268}N^{182} - 8L^{267}N^{182})D^{447} +$$
$$+(75L^{270}N^{186} - L^{269}N^{185} - 292L^{269}N^{184} + 24L^{269}N^{183} + L^{268}N^{186} -$$
$$-201L^{268}N^{184} - 64L^{268}N^{183} - 28L^{267}N^{184} + 292L^{267}N^{181} +$$

$$+200L^{266}N^{181}+28L^{265}N^{181})D^{446}+\cdots+(380L^{34}N^{30}-45L^{34}N^{29}+$$
$$+189L^{34}N^{28}-26L^{34}N^{27}+10L^{34}N^{26}-49L^{33}N^{29}+377L^{33}N^{28}-$$
$$-1210L^{33}N^{27}+481L^{33}N^{26}-465L^{33}N^{25}-540L^{32}N^{28}+28L^{32}N^{27}-$$
$$-88L^{32}N^{26}-253L^{32}N^{25}+195L^{32}N^{24}+50L^{31}N^{27}-67L^{31}N^{26}+$$
$$1244L^{31}N^{25}-112L^{31}N^{24}+220L^{31}N^{23}+1320L^{31}N^{21}+55L^{30}N^{26}-$$
$$-11L^{30}N^{25})D^{63}+(5L^{33}N^{26}-95L^{33}N^{28}-25L^{33}N^{27}-211L^{33}N^{29}+$$
$$+50L^{32}N^{28}-142L^{32}N^{27}+997L^{32}N^{26}-5L^{32}N^{25}+110L^{32}N^{24}+$$
$$+20L^{31}N^{27}+7L^{31}N^{26}-106L^{31}N^{25}+140L^{31}N^{24}-1140L^{31}N^{23}-$$
$$-10L^{30}N^{26}+242L^{30}N^{24}+22L^{30}N^{23})D^{62}+(20L^{32}N^{27}-49L^{32}N^{28}-$$
$$-20L^{32}N^{26}-2L^{31}N^{27}-11L^{31}N^{26}-12L^{31}N^{25}+55L^{30}N^{26}-5L^{30}N^{25}+$$
$$+22L^{30}N^{24}+220L^{30}N^{22})D^{61}+(50L^{31}N^{27}+5L^{31}N^{25}-5L^{30}N^{26}+$$
$$+7L^{30}N^{25}-112L^{30}N^{24}-11L^{29}N^{25}+L^{29}N^{24})D^{60}+(L^{29}N^{25}-2L^{30}N^{26}+$$
$$+22L^{29}N^{23})D^{59}-5L^{29}N^{25}D^{58}+L^{28}N^{24}D^{57},$$
$$T_{den}(D,L,N)=(L^{270}N^{186}-L^{268}N^{183})D^{446}+(L^{269}N^{187}-L^{270}N^{185}-$$
$$-25L^{269}N^{185}+L^{269}N^{184}-8L^{268}N^{185}+25L^{267}N^{182}+8L^{266}N^{182})D^{445}+$$
$$+(25L^{269}N^{184}-75L^{269}N^{186}-25L^{268}N^{186}+L^{268}N^{185}+300L^{268}N^{184}-$$
$$-24L^{268}N^{183}-9L^{267}N^{186}+201L^{267}N^{184}+64L^{267}N^{183}+28L^{266}N^{184}-$$
$$-292L^{266}N^{181}-200L^{265}N^{181}-28L^{264}N^{181})D^{445}+\cdots+$$
$$+(49L^{31}N^{28}-20L^{31}N^{27}+20L^{31}N^{26}+2L^{30}N^{27}+11L^{30}N^{26}+12L^{30}N^{25}-$$
$$-55L^{29}N^{26}+5L^{29}N^{25}-22L^{29}N^{24}-220L^{29}N^{22})D^{59}+(5L^{29}N^{26}-5L^{30}N^{25}-$$
$$-50L^{30}N^{27}-75L^{29}N^{25}+112L^{29}N^{24}+11L^{28}N^{25}-L^{28}N^{24})D^{58}+$$
$$+(2L^{29}N^{26}-L^{28}N^{25}-22L^{28}N^{23})D^{58}+5L^{23}N^{20}D^{51}-L^{22}N^{20}D^{50}. \qquad (3.457)$$

Тепер, розглядаючи, як і в прикладі 3.69, лише останні доданки записаних співвідношень (3.457), що відповідають найменшим степеням змінної *D*, для передавальної функції скінченного автомату можна записати наступний спрощений аналітичний вираз:

$$T(D,L,N)>\frac{L^{28}N^{24}D^{57}}{5L^{23}N^{20}D^{51}-L^{22}N^{20}D^{50}}=\frac{L^{6}N^{4}D^{7}}{5LND-1}=$$

$$= -L^6N^4D^7(1 + 5LND + 25L^2N^2D^2 + 125L^3N^3D^3 + \cdots) =$$

$$-L^6N^4D^7 - 5L^7N^5D^8 - 25L^8N^6D^9 - 125L^9N^7D^{10} - \cdots \qquad (3.458)$$

Тобто, результати проведених аналітичних та комп'ютерних розрахунків показують, що, згідно з остаточним співвідношенням (3.458), для згорткового коду з параметрами $K = 5$, $\frac{1}{n} = \frac{1}{3}$, який розглядається, параметр мінімального просвіту становить $d_{min} = 7$. Цій кодовій відстані на ґратковій діаграмі відповідає шлях із шістьма гілками, а чотири з цих гілок описують вхідні сигнали із значенням 1.

3.8.13.5 Порівняльний аналіз методів ручного та комп'ютерного розрахунку мінімального просвіту згорткового коду

У попередніх підрозділах було розглянуто декілька прикладів розрахунку передавальної функції для декодерів згорткового коду. В результаті, за аналітичним виразом для дрібно-раціональної передавальної функції, був розрахований параметр мінімального просвіту D для різних згорткових кодів з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$; $K = 4$, $\frac{1}{n} = \frac{1}{3}$ та $K = 5$, $\frac{1}{n} = \frac{1}{3}$. Для простих кодів із невеликою надлишковістю, розглянутих у підрозділі 3.8.13.3 та у прикладі 3.69, наведеному у підрозділі 3.8.13.4, тобто, для кодів з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$ та $K = 4$, $\frac{1}{n} = \frac{1}{3}$, всі алгебраїчні операції над поліномами були проведені ручним способом. У той самий час, для більш складного коду з параметрами $K = 5$, $\frac{1}{n} = \frac{1}{3}$, вручну були проведені лише початкові розрахунки, які дозволили встановити головні зв'язки між станами автомату та оцінити степені поліноміальних функцій. Подальші розрахунки, пов'язані з перетворенням поліномів, проводились на персональному комп'ютері з використанням засобів символьного процесора системи науково-технічних розрахунків MatLab, призначених для роботи з поліномами. Розглянуті вище приклади 3.69 та 3.70 дозволяють зробити порівняльний аналіз ручних та комп'ютерних методів розрахунку, які можуть

бути використані для формування передавальної функції скінченного автомату та її подальшого аналізу з метою визначення мінімального просвіту згорткового коду.

У будь-якому разі зрозуміло, що методи ручних розрахунків можна застосовувати лише для простих кодувальних систем, в яких кількість станів скінченного автомату не перевищує 10. За такої умови максимальна степінь поліномів не перевищує 20, а кількість елементарних операцій додавання та віднімання, які виконуються для зведення подібних доданків під час пошуку коефіцієнтів поліному, є меншою за 100. Тобто, хоча такі аналітичні перетворення є досить громіздкими, їх можна виконати і вручну. З іншого боку, суттєвою перевагою ручного способу проведення поліноміальних операцій є можливість пошуку більш компактного запису відповідних коефіцієнтів з їхнім групуванням за кількома змінними. Так, всі поліноми в прикладах 3.69 та 3.70 були записані відносно змінної D, але групування проводилось також за змінними L та N. Наприклад, аналітичний вираз $D^{14}(L^{10}N^3 + L^8N^2 - L^3N - 3)$ може бути переписаний більш компактно, якщо у множнику, який стоїть у дужках, додатково згрупувати всі доданки за степенями змінних L та N. Відповідні аналітичні перетворення мають наступний вигляд:

$$D^{14}(L^{10}N^3 + L^8N^2 - L^3N - 3) = D^{14}(L^3N(L^7N^2 + L^5N - 1) - 3) =$$
$$= D^{14}(L^3N(L^5N(L^2N + 1) - 1) - 3).$$

Зрозуміло, що математичні вирази з великою кількістю дужок іноді важко читати та аналізувати, але, з іншого боку, групування поліноміальних виразів за кількома змінними не лише дає більш компактну форму запису та знижує степені допоміжних змінних, але й у багатьох випадках дозволяє встановити важливі додаткові закономірності, пов’язані з особливостями роботи скінченного автомату. Тому у прикладах 3.69 та 3.70 аналітичні вирази для станів скінченного автомату (3.381) – (3.396) та (3.421) – (3.442) записані у компактній формі. Тобто, в поліномах, всі доданки яких впорядковані відносно змінної D, поліноміальні

коефіцієнти подані не у вигляді інших поліномів відносно змінних L та N, а, у разі наявності такої можливості, розкладені на окремі множники. Навпаки, символьний процесор системи науково-технічних розрахунків MatLab завжди подає результати алгебраїчних дій над поліномами таким способом, що поліноміальні коефіцієнти також є поліномами відносно інших змінних. В цьому можна остаточно впевнитись, аналізуючи результати аналітичних розрахунків, які наведені у додатку Р. Це є однією з причин, за якою аналітичні вирази, отримані в результаті комп'ютерних розрахунків, зазвичай є непомірно громіздкими. Проте, у будь-якому разі, символьний процесор працює правильно та безпомилково. Якщо дослідник бажає мати не поліноміальну, а якусь іншу форму подання для деяких математичних виразів, для цього можуть бути використані інші функції символьного процесора [13, 14].

Іншою суттєвою перевагою ручного способу розрахунку поліноміальних коефіцієнтів є можливість введення допоміжних змінних у разі непомірного ускладнення аналітичних виразів. Такий спосіб запису аналітичних співвідношень з метою їх спрощення та подальшого аналізу також вкрай ефективно використовувався у прикладах 3.69 та 3.70.

Аналіз прикладу 3.70 також показує, що ручні розрахунки бажано використовувати на етапі загальної постановки завдання та пошуку взаємозв'язків між станами автомату. Крім цього, вони є вкрай ефективними під час проведення подальших розрахунків, коли здійснюється формування та аналіз простих аналітичних виразів та пошук залежностей між поліномами у загальній формі. З іншого боку, для виконання громіздких алгебраїчних операцій з поліномами високих порядків, вищих за 20, зручніше та ефективніше використовувати можливості сучасного програмного забезпечення, безпосередньо призначеного для виконання аналітичних розрахунків, наприклад, символьного процесора MatLab. Слід відзначити, що навіть для сучасних швидкодійних комп'ютерних систем групування поліноміальних

коефіцієнтів за степенями та зведення подібних доданків за умови степені поліномів, вищої за 200, є досить складним обчислювальним завданням, виконання якого потребує певного часу. Наприклад, під час виконання завдання для прикладу 3.70, загальний час розв'язування задач пошуку залежності $X_i(X_h)$ та передавальної функції скінченного автомату $T(D, L, N)$, заданих аналітичними співвідношеннями (3.455) та (3.457), (3.458), на персональному комп'ютері з процесором Intel Xenon з частотою 2,83 МГц та об'ємом оперативної пам'яті 8 Гб мав складати кілька діб. Тому така постановка завдання для системи комп'ютерних розрахунків вважається некоректною, оскільки час проведення розрахунків перевищує критичну величину, яка для сучасних комп'ютерних систем має бути у межах кількох десятків хвилин [21, 22]. Головною причиною складності виконання алгебраїчних операцій з поліномами високих порядків є непомірне зростання елементарних арифметичних дій за умови збільшення порядку поліноміальної функції. Для двох поліномів порядку n кількість операцій поелементного множення на початковому етапі розкриття дужок складає n^2, а завдання пошуку доданків з однаковим степенем поліноміальної змінної, пов'язане із завданням перебирання великої кількості отриманих елементарних доданків, взагалі є одним з найскладніших завдань комп'ютерної арифметики [31, 56]. Сьогодні найбільш ефективним інструментом для розв'язування подібних завдань підвищеної обчислювальної складності є проведення хмарних обчислень в глобальних комп'ютерних мережах з розпаралелюванням обчислювального процесу, що дозволяє використовувати великі обчислювальні потужності віддалених серверів [21, 22].

Обійти цю проблему загальної постановки завдання проведення аналітичних обчислень з використанням символьного процесора для прикладу, який розглядається, можна наступним чином. Із співвідношення (3.455) зрозуміло, що складність обчислення коефіцієнтів поліному для визначення

функціональної залежності $X_f(D,\ L,\ N,\ X_h)$ обумовлена складністю полінома $X_e(D, L, N, X_h)$, який використовується для проведення обчислень. Саме велика кількість доданків в коефіцієнтах поліному $X_e(D, L, N, X_h)$ робить такі обчислення непомірно складними. З іншого боку, нам необхідно знати лише ті коефіцієнти, які стоять перед старшими та молодшими степенями поліномів $T_{nom}(D,\ L,\ N)$ та $T_{den}(D,\ L,\ N)$. Відповідно, і в поліномі $X_f(D, L, N, X_h)$ нас цікавлять лише ті коефіцієнти, які стоять біля старших та молодших степенів змінної D, а інші коефіцієнти можна взагалі не розглядати. За таких умов проведення аналітичних операцій з поліномами з використанням символьного процесору значно спрощується. Для проведення аналізу поліномів $X_{enom}(D,\ L,\ N,\ X_h)$ та $X_{eden}(D, L, N, X_h)$ вони були записані до файлів **`xenom.txt`** та **`xeden.txt`**, і в результаті аналізу інформації, записаної до цих файлів, формуються скорочені поліноми X_{enom}^{short} та X_{eden}^{short}. У загальному випадку спосіб формування скорочених поліномів X_{snom}^{short} та X_{sden}^{short} для будь-якого стану скінченного автомату X_s полягає в тому, що з поліному, записаного в повній формі, відкидаються всі доданки, крім тих, яким відповідають високі та низькі степені змінної D. З точки зору формалізму дискретної математики [48], для чисельника та знаменника функції стану автомату X_s, тобто для поліномів $X_{snom}(D,\ L,\ N,\ X_h)$ та $X_{sden}(D,\ L,\ N,\ X_h)$, записаних у загальній формі, скорочена поліноміальна форма $X_{snom}^{short}(D, L, N, X_h)$ та $X_{eden}^{short}(D, L, N, X_h)$ записується у вигляді наступних математичних співвідношень:

$$X_{snom}^{short}(D, L, N, X_h) = \left(\sum_{i=n_{\max}^{snom}-k_{snom}}^{n_{\max}^{snom}} K_{inom}(L, N) D^i + \right.$$

$$\left. + \sum_{j=n_{\min}^{snom}}^{n_{\min}^{snom}+l_{snom}} K_{jnom}(L, N) D^j \right) X_h, \tag{3.459}$$

$$X_{eden}^{short}(D, L, N, X_h) = \left(\sum_{i=n_{\max}^{den}-k_{sden}}^{n_{\max}^{den}} K_{iden}(L, N) D^i + \right.$$

$$+\sum_{j=n_{\min}^{den}}^{n_{\min}^{den}+l_{sden}} K_{jden}(L,N)D^j\Bigg)X_h,$$

де $n_{\max}^{snom}$ – значення максимальної степені змінної D у поліномі чисельника $X_{snom}(D, L, N, X_h)$, $n_{\max}^{sden}$ – значення максимальної степені змінної D у поліномі знаменника $X_{sden}(D, L, N, X_h)$, $n_{\min}^{snom}$ – значення мінімальної степені змінної D у поліномі чисельника $X_{snom}(D, L, N, X_h)$, $n_{\min}^{sden}$ – значення мінімальної степені змінної D у поліномі знаменника $X_{sden}(D, L, N, X_h)$, k_{snom} – кількість складових поліному $X_{snom}(D, L, N, X_h)$ за максимальними степенями, l_{snom} – кількість складових поліному $X_{snom}(D, L, N, X_h)$ за мінімальними степенями, k_{sden} – кількість складових поліному $X_{sden}(D, L, N, X_h)$ за максимальними степенями, l_{sden} – кількість складових поліному $X_{sden}(D, L, N, X_h)$ за мінімальними степенями, K_{inom}, K_{jnom}, K_{iden}, та K_{jden} – поліноміальні коефіцієнти для доданків із степенями i та j відповідно для функцій $X_{snom}(D, L, N, X_h)$ та $X_{sden}(D, L, N, X_h)$.

Результати розрахунків поліноміальних коефіцієнтів для станів скінченного автомату $X_e(D, L, N, X_h)$ та $X_i(D, L, N, X_h)$ з використанням співвідношень (3.453), (3.455), (3.459) наведені у додатку Р. Для стану автомату X_i поліном чисельника $X_{inom}(D, L, N)$, обчислений з використанням алгебраїчних операцій над поліномами та функцій символьного процесора **`expand()`** та **`collect()`**, записувався до текстового файлу **`xinom.txt`**, а поліном знаменника $X_{inom}(D, L, N)$ – відповідно до файлу **`xiden.txt`**. Вміст отриманих в результаті проведених розрахунків текстових файлів **`xenom.txt`**, **`xeden.txt`**, **`xinom.txt`** та **`xiden.txt`** з початковими та кінцевими доданками відповідних поліномів наведений у додатку Р.

Зрозуміло, що якщо максимальна та мінімальна степені поліномів $X_{enom}(D, L, N, X_h)$, $X_{eden}(D, L, N, X_h)$, $X_{inom}(D, L, N, X_h)$ та $X_{iden}(D, L, N, X_h)$ є результатом попередніх розрахунків за співвідношеннями (3.439) – (3.452), то значення l_{enom}, l_{eden}, k_{enom}, k_{eden}, l_{inom}, l_{iden}, k_{inom} та k_{iden}, які задають обмеження на кількість складових у створюваних поліномах $X_{enom}^{short}(D, L, N, X_h)$, $X_{eden}^{short}(D, L, N, X_h)$,

$X_{inom}^{short}(D, L, N, X_h)$, $X_{iden}^{short}(D, L, N, X_h)$, обираються залежно від складності поставленого обчислювального завдання та потужності використовуваної комп'ютерної системи. Для завдання, яке розв'язувалося, були обрані наступні параметри функцій станів $X_e(D, L, N, X_h)$ та $X_i(D, L, N, X_h)$: $l_{enom} = 10$, $k_{enom} = 13$, $l_{eden} = 9, k_{enom} = 13$; $l_{inom} = 11, k_{inom} = 13, l_{iden} = 19, k_{inom} = 21$. Відповідно, максимальна та мінімальна степені змінної D для поліномів $X_{enom}(D, L, N, X_h)$, $X_{eden}(D, L, N, X_h)$, $X_{inom}(D, L, N, X_h)$ та $X_{iden}(D, L, N, X_h)$, коефіцієнти яких були обчислені за співвідношеннями (3.439) – (3.452) складали: $n_{\max}^{enom} = 239$, $n_{\min}^{enom} = 14$, $n_{\max}^{eden} = 234, n_{\min}^{eden} = 42$ та $n_{\max}^{inom} = 343, n_{\min}^{inom} = 35, n_{\max}^{iden} = 335, n_{\min}^{iden} = 60$.

Як можна побачити за результатами проведеного аналізу та наведених прикладів, загалом складність завдань дискретної математики, комбінаторного аналізу та, безпосередньо пов'язаної з дискретною математикою, теорії скінченних автоматів, полягає не в складності самих базових систем рівнянь, які зазвичай є лінійними та простими, а у великій кількості варіантів, які необхідно прорахувати з метою пошуку оптимального з них [48 – 50]. Саме таким і є завдання пошуку мінімального просвіту згорткового коду, яке безпосередньо пов'язано з теорією скінченних автоматів [33, 62, 63]. Зрозуміло, що складність цього завдання та необхідна кількість поліноміальних операцій насамперед залежить від складності коду та від його параметрів. Розглянутий у прикладі 3.70 згортковий код з параметрами $K = 5$, $\frac{1}{n} = \frac{1}{3}$ є кодом середньої степені складності, а згорткові коди з параметрами $K = 4, \frac{1}{n} = \frac{1}{3}$ та $K = 3, \frac{1}{n} = \frac{1}{2}$ вважаються простими, тому саме вони часто використовуються в навчальній літературі для вивчення загальних способів формування таких кодів [33, 62, 63]. Сьогодні більшість складних та ефективних конструкцій згорткових кодів переважно розробляються через проведення розрахунків на швидкодійних комп'ютерних системах з використанням відповідних сучасних програмних засобів, в яких реалізовані обчислювальні методи дискретної математики, комбінаторного аналізу, теорії скінченних автоматів та теорії кодування [33, 52 – 56, 62, 63].

3.8.13.6 Залежність коректувальної здатності згорткового коду від передавальної функції та параметру мінімального просвіту

У літературі з теорії кодування показано, що за умови відомої передавальної функції декодеру згорткового коду $T(D, L, N)$, у разі обрання жорсткої схеми прийняття рішень, яка відповідає двійковим цифровим кодам, максимальне значення імовірності бітової помилки у кодовій комбінації P_B може бути розрахована наступним чином [33, 62, 63]:

$$P_B \leq \left.\frac{dT(D,N)}{dN}\right|_{N=1,D=2\sqrt{p(1-p)}}, \tag{3.460}$$

де p – імовірність бітової помилки у поодиночному символі.

Наприклад, для коду з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$, передавальна функція якого розглядалась в підрозділі 3.8.13.4, згідно із співвідношенням (3.373) можна записати:

$$T(D,L,N)|_{L=1} = \left.\frac{D^5L^3N}{1-DLN(1+L)}\right|_{L=1} = \frac{D^5N}{1-2DN}.$$

Тоді:

$$\left.\frac{dT(D,N)}{dN}\right|_{N=1} = \left.\frac{D^5(1-2DN)+2DD^5}{(1-2DN)^2}\right|_{N=1} = \frac{D^5-2D^6+2D^6}{(1-2D)^2} =$$

$$= \frac{D^5}{(1-2D)^2}. \tag{3.461}$$

Враховуючи те, що, згідно із співвідношенням (3.460), $D = 2\sqrt{p(1-p)}$, а також отримане співвідношення (3.461) для похідної $\left.\frac{dT(D,N)}{dN}\right|_{N=1}$, для максимального значення імовірності бітової помилки у комбінації згорткового коду з параметрами $K = 3, \frac{1}{n} = \frac{1}{2},$ можна записати натупний аналітичний вираз [33]:

$$P_B(p) \leq \left.\frac{dT(D,N)}{dN}\right|_{N=1,D=2\sqrt{p(1-p)}} = \left.\frac{D^5}{(1-2D)^2}\right|_{D=2\sqrt{p(1-p)}} = \frac{\left(2\sqrt{p(1-p)}\right)^5}{\left(1-4\sqrt{p(1-p)}\right)^2}. \tag{3.462}$$

Графічна залежність $P_B(p)$, отримана з використанням аналітичного співвідношення (3.462) для згорткового коду з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$, наведена на рис. 3.109. З наведеної залежності зрозуміло, що мінімальна величина максимального значення імовірності помилки у згортковому коді складає $P_B = 0{,}905$ та досягається за умови $p = 0{,}215$ та $p = 0{,}785$. Максимальна величина цього значення відповідає умові $p = 0{,}5$ та становить $P_B = 1$. Функція $P_B(p)$, задана співвідношенням (3.462), є симетричною відносно лінії $p = 0{,}5$, тобто, завжди виконується умова $P_B(p) = P_B(1 - p)$.

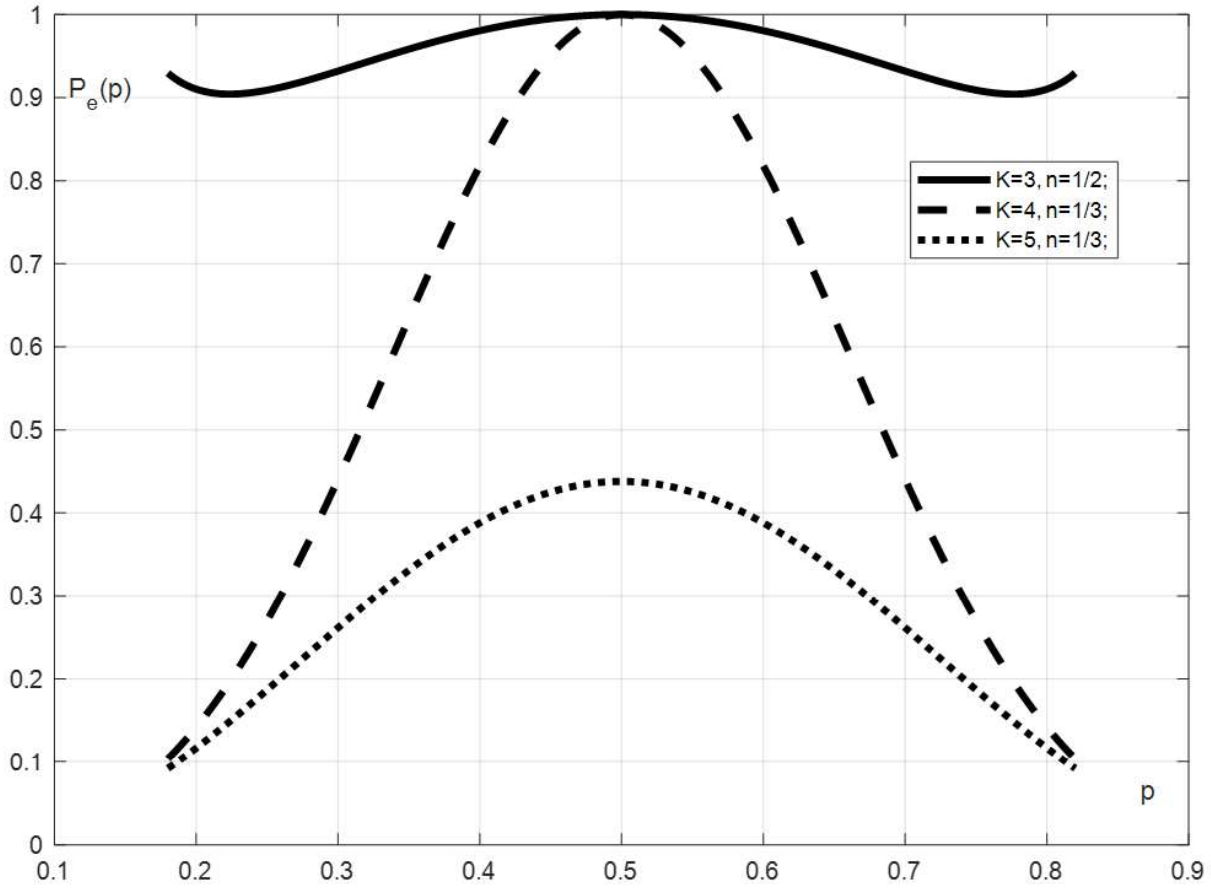


Рис. 3.109 Залежності максимального значення імовірності помилки у різних типах згорткових кодів від імовірності бітової помилки $P_e(p)$ за умови застосування жорсткої схеми кодування, отримані з використанням аналітичних співвідошень (3.462), (3.465) та (3.468)

Для згорткового коду з параметрами $K = 4, \frac{1}{n} = \frac{1}{3}$ передавальна функція визначається співвідношенням (3.400). Перепишемо її для окремого випадку $L = 1$ наступним чином:

$$T(D,N)|_{L=1} = \frac{D^6L^4N^2}{1-D^2L^2N^2}\Big|_{L=1} = \frac{D^6N^2}{1-D^2N^2}. \quad (3.463)$$

Відповідно, аналітичний вираз для похідної функції (3.463) $T(D, N)$, який необхідно підставити у співвідношення (3.460) для визначення максимальної імовірності помилки у згортковому коді, має наступний вигляд:

$$\frac{dT(D,N)}{dN}\Big|_{N=1} = \frac{2D^6N(1-D^2N^2)+2D^2ND^6N^2}{(1-D^2N^2)^2}\Big|_{N=1} =$$

$$= \frac{2D^6N-2D^8N^3+2D^8N^3}{(1-D^2N^2)^2}\Big|_{N=1} = \frac{2D^6}{(1-D^2)^2}. \quad (3.464)$$

З урахуванням співвідношень (3.460) та (3.464), для максимальної величини імовірності помилки у згортковому коді з параметрами $K = 4, \frac{1}{n} = \frac{1}{3}$, отримуємо нступний аналітичний вираз:

$$P_B(p) \le \frac{dT(D,N)}{dN}\Big|_{N=1,D=2\sqrt{p(1-p)}} = \frac{64\big(p(1-p)\big)^3}{\big(1-4p(1-p)\big)^2},$$

або

$$P_B(p) < \frac{32\big(p(1-p)\big)^3}{\big(1-2p(1-p)\big)^2}. \quad (3.465)$$

Графічна залежність $P_B(p)$, отримана з використанням аналітичного співвідношення (3.465) для згорткового коду з параметрами $K = 4, \frac{1}{n} = \frac{1}{3}$, також наведена на рис. 3.109. Зрозуміло, що для коду з такими параметрами функція $P_B(p)$ також є симетричною відносно лінії $p = 0{,}5$, для цього значення $P_B(0{,}5) = 1$. Для коду з такими параметрами характер функції $P_B(p)$ в усьому діапазоні значень p відповідає розподілу Гаусса, добре відомому з основ теорії імовірностей [49].

Для передавальної функції згорткового коду з параметрами $K = 5, \frac{1}{n} = \frac{1}{3}$, існує аналітичний вираз, заданий співвідношенням (3.458). Спочатку, як і для коду з параметрами $K = 4, \frac{1}{n} = \frac{1}{3}$, перепишемо співвідношення (3.458) для окремого випадку $L = 1$:

$$T(D,N)|_{L=1} = \frac{N^4 D^7}{5ND-1}. \quad (3.466)$$

Для похідної $\frac{dT(D,N)}{dN}\Big|_{N=1}$ передавальної функції $T(D,N)|_{L=1}$ згорткового коду з параметрами $K=4$, $\frac{1}{n}=\frac{1}{3}$, заданої співвідношенням (3.466), можна записати наступний аналітичний вираз:

$$\frac{dT(D,N)}{dN}\Big|_{N=1} = \frac{3N^3D^7(5ND-1)-5DN^4D^7}{(5ND-1)^2}\Big|_{N=1} =$$

$$= \frac{15N^4D^8-3N^3D^7-5N^4D^8}{(5ND-1)^2}\Big|_{N=1} = \frac{10N^4D^8-3N^3D^7}{(5ND-1)^2}\Big|_{N=1} =$$

$$= D^7\frac{10D-3}{(5D-1)^2}. \quad (3.467)$$

Тоді, враховуючи отримане аналітичне співвідношення (3.467) та узагальнений вираз для максимального значення імовірності помилки у згортковому коді (3.460), для коду з параметрами $K = 5, \frac{1}{n}=\frac{1}{3}$, остаточно можна записати:

$$P_B(p) \le \frac{dT(D,N)}{dN}\Big|_{N=1,D=2\sqrt{p(1-p)}} = D^7\frac{10D-3}{(5D-1)^2}\Big|_{D=2\sqrt{p(1-p)}} =$$

$$= \left(2\sqrt{p(1-p)}\right)^7 \frac{20\sqrt{p(1-p)}-3}{\left(10\sqrt{p(1-p)}-1\right)^2} =$$

$$= 128\left(\sqrt{p(1-p)}\right)^7 \frac{20\sqrt{p(1-p)}-3}{\left(10\sqrt{p(1-p)}-1\right)^2} =$$

$$= \frac{2560\left(p(1-p)\right)^4-384\left(p(1-p)\right)^3\sqrt{p(1-p)}}{\left(10\sqrt{p(1-p)}-1\right)^2}. \quad (3.468)$$

На рис. 3.109 також наведена графічна залежність $P_B(p)$, отримана з використанням аналітичного співвідношення (3.467) для згорткового коду з параметрами $K=5, \frac{1}{n}=\frac{1}{3}$. У цьому випадку функція $P_B(p)$ також є симетричною відносно лінії p = 0,5, але найбільша величина максимальної імовірності помилки у такому коді становить $P_B(0,5)$ = 0,4375. Тобто, на відміну від інших розглянутих кодів, максимальна величина бітової помилки у згортковому коді з параметрами $K = 5, \frac{1}{n}=\frac{1}{3}$, завжди є значно меншою за 1 та не перевищує

значення 0,5.

Крім оцінок імовірності помилки у згортковому коді для імпульсних двійкових сигналів за умови жорсткої схеми прийняття рішень, існують також відповідні аналітичні оцінки для інших умов поширення сигналів у телекомунікаційних системах, зокрема, для синусоїдальних сигналів з кодоімпульсною модуляцією [1, 33, 49, 62, 63].

Наприклад, для каналу з гауссовим шумом за умови фазової кодоімпульсної модуляції двійкового сигналу імовірність помилки у згортковому коді оцінюється наступним чином [33, 62, 63]:

$$P_B\left(\frac{E_c}{N_0}\right) \le \mathrm{Q}\left(\sqrt{2d_f\frac{E_c}{N_0}}\right)\exp\left(d_f\frac{E_c}{N_0}\right)\left.\frac{dT(D,N)}{dN}\right|_{N=1,D=\exp\left(-\frac{E_c}{N_0}\right)}, \qquad (3.469)$$

де $\frac{E_b}{N_0}$ – відношення енергії інформаційного сигналу E_b до густини спектру потужності шуму N_0, $\frac{E_c}{N_0} = \frac{rE_b}{N_0}$ – відношення загальної енергії канального символу до густини спектру потужності шуму N_0, $r = k / n$ – степінь кодування, або надлишковість згорткового коду, d_f – параметр просвіту згорткового коду, $\mathrm{Q}(x)$ – добре відома з основ теорії ймовірностей функція помилок Гаусса, або інтегральна функція помилок [1, 49]:

$$\mathrm{Q}(x) = \frac{1}{\sqrt{2\pi}}\int_x^\infty \exp\left(-\frac{u^2}{2}\right)du. \qquad (3.470)$$

Для оцінки імовірності помилки у згортковому коді з використанням співвідношень (3.469), (3.470), використовуються аналітичні вирази (3.461), (3.464) та (3.467) для похідної $\left.\frac{dT(D,N)}{dN}\right|_{N=1}$. Відмінність оцінок, які проводяться з використанням співвідношення (3.469) полягає у тому, що в аналітичні вирази (3.461), (3.464) та (3.467) замість змінної D, згідно зі співвідношенням (3.469), підставляється значення $D = \exp\left(-\frac{E_c}{N_0}\right)$.

Наприклад, для згорткового коду з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$, згідно зі співвідношеннями (3.461), (3.469), враховуючи, що $d_f = 5$, можна записати:

$$P_B\left(\frac{E_b}{N_0}\right) \le \mathrm{Q}\left(\sqrt{\frac{5E_b}{N_0}}\right)\exp\left(\frac{5E_b}{2N_0}\right)\frac{\exp\left(-\frac{5E_b}{2N_0}\right)}{\left(1-2\exp\left(-\frac{E_b}{2N_0}\right)\right)^2} =$$

$$= \frac{\mathrm{Q}\left(\sqrt{\frac{5E_b}{N_0}}\right)}{\left(1-2\mathrm{ex}\ \left(-\frac{E_b}{2N_0}\right)\right)^2}. \quad (3.471)$$

Відповідно, для коду з параметрами $K = 4, \frac{1}{n} = \frac{1}{3}$, згідно зі співвідношеннями (3.464), (3.469), та враховуючи, що $d_f = 6$, можна записати:

$$P_B\left(\frac{E_b}{N_0}\right) \le \mathrm{Q}\left(\sqrt{\frac{4E_b}{N_0}}\right)\exp\left(\frac{2E_b}{N_0}\right)\frac{2\exp\left(-\frac{2E_b}{N_0}\right)}{\left(1-\exp\left(-\frac{2E_b}{3N_0}\right)\right)^2} =$$

$$= \frac{\mathrm{Q}\left(\sqrt{\frac{4E_b}{N_0}}\right)}{\left(1-\exp\left(-\frac{2E_b}{3N_0}\right)\right)^2}. \quad (3.472)$$

Для коду з параметрами $K = 5, \frac{1}{n} = \frac{1}{3}$, згідно зі співвідношеннями (3.467), (3.469), та враховуючи, що $d_f = 7$, відповідно, маємо:

$$P_B\left(\frac{E_b}{N_0}\right) \le \mathrm{Q}\left(\sqrt{\frac{14E_b}{3N_0}}\right)\exp\left(\frac{7E_b}{3N_0}\right)\exp\left(-\frac{7E_b}{3N_0}\right)\frac{10\exp\left(-\frac{E_b}{3N_0}\right)-3}{\left(5\exp\left(-\frac{E_b}{3N_0}\right)-1\right)^2} =$$

$$= \mathrm{Q}\left(\sqrt{\frac{14\ \ b}{3N_0}}\right)\frac{10\exp\left(-\frac{E_b}{3N_0}\right)-3}{\left(5\exp\left(-\frac{E_b}{3N_0}\right)-1\right)^2}. \quad (3.473)$$

Залежності $P_B\left(\frac{E_b}{N_0}\right)$, отримані з використанням співвідношень (3.471) – (3.473), наведені на рис. 3.110. Зрозуміло, що загалом із збільшенням співвідношення потужності сигналу до потужності шуму зменшується імовірність помилки у каналі зв'язку. Але закономірність зменшення імовірності помилки у згортковому коді у разі покращення коректувальної здатності коду у даному випадку не спостерігається, на відміну від оцінок імовірності помилки для жорсткої схеми кодування, наведених на рис. 3.109. Це насамперед пов'язано з тим, що оцінки з використанням співвідношень (3.471) – (3.473) є наближеними та зазвичай дещо завищеними. Тобто, оцінюються максимальне значення імовірності похибки, а реальне значення майже завжди є нижчим. Наприклад, згідно з залежністю, наведеною на рис. 3.110, мінімальне значення максимальної величини імовірності помилки у згортковому коді з параметрами $K = 4, \frac{1}{n} = \frac{1}{3}$,

навіть у разі високих значень співвідношення потужності сигналу до потужності шуму, складає 0,4, проте реально це значення може бути близьким до 0. Аналогічний висновок можна зробити відносно коду з параметрами $K = 5$, $\frac{1}{n} = \frac{1}{3}$, для якого за умови зростання співвідношення потужності сигналу до потужності шуму максимальне значення імовірності помилки у коді дещо зростає від значення 0,09 до 0,2.

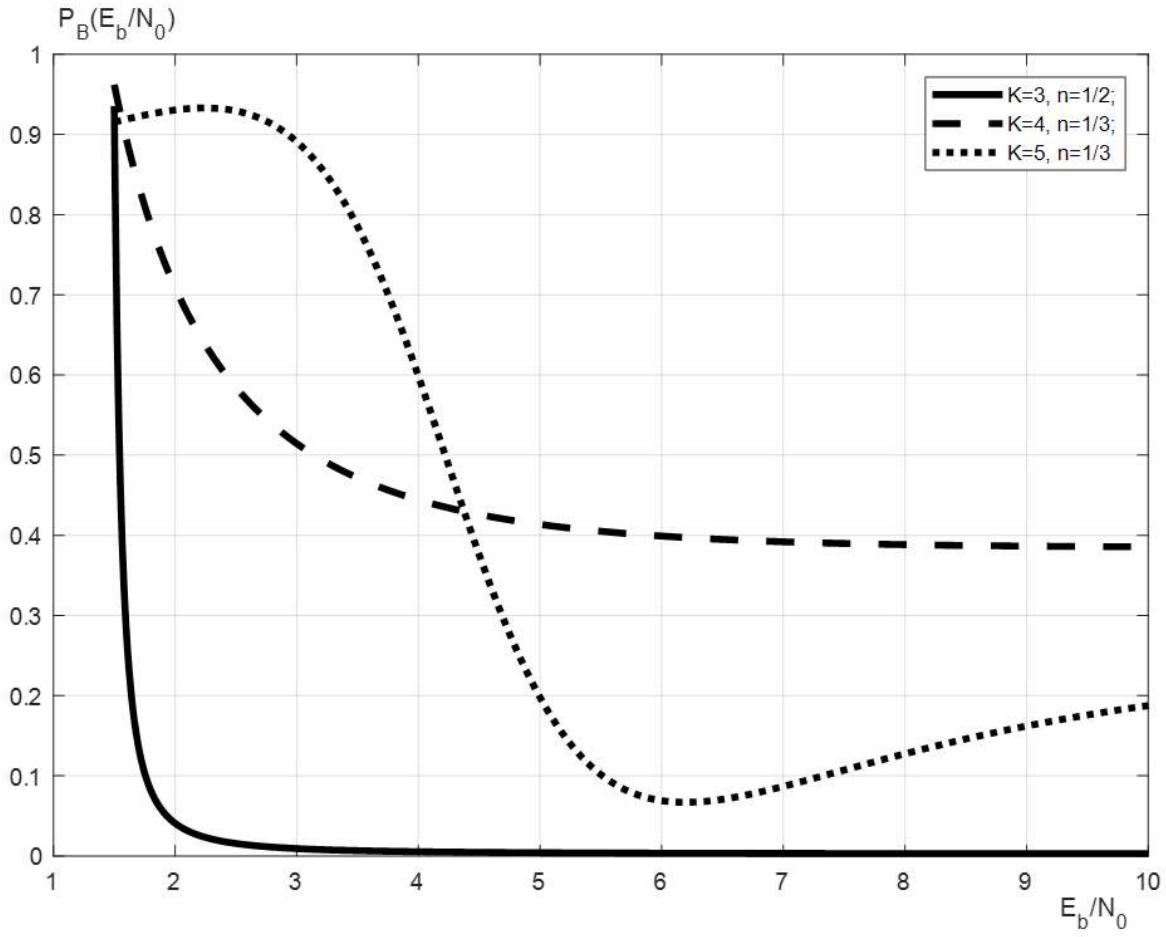


Рис. 3.110 Залежності максимального значення імовірності помилки у різних типах згорткових кодів від співвідношення потужності сигналу до потужності шуму $P_B\left(\frac{E_b}{N_0}\right)$ за умови поширення двійкового сигналу з фазовою кодоімпульсною модуляцією у каналі зв'язку з гауссовим шумом, отримані з використанням аналітичних співвідошень (3.471) – (3.473)

У будь-якому разі, теоретичні оцінки передавальної функції згорткових кодів, проведені у підрозділі 3.8.13.4 з використанням теорії скінченних автоматів та методів дискретної математики, а також отримані на основі цих функцій

аналітичні співвідношення (3.461) – (3.465) для оцінки максимальної величини імовірності помилки у згортковому коді за умови жорсткого прийняття рішень, є правильними. Щодо співвідношень (3.471) – (3.473), тут завищені оцінки мінімального значення імовірності помилки у каналі зв'язку з гауссовим шумом обумовлені складністю формування більш точних математичних моделей для таких фізичних умов поширення цифрового сигналу [1].

Із проведених теоретичних оцінок зроузміло, що у разі збільшення параметрів згорткового коду, як-то довжини кодового обмеження K та коефіцієнту надлишковості n, поліпшується його коерктувальна здатність, особливо за умови низького значення співвідношення потужності сиганлу до потужності шуму. Проте, з іншого боку, збільшення значень параметрів K та n веде до зменшення продуктивності приймально-передавальної системи та до її ускладнення. Тобто, для реальних систем зв'язку пошук оптимальних значень K та n базується на оцінках ефетивності кодування з використанням співвідношень (3.460) – (3.473) та на інформаційних та економічних оцінках всієї приймально-передавальної системи [1, 33, 49, 62, 63]. У будь-якому разі, пошук передавальної функції згорткового коду з відповідними параметрами $T(D, N, L)$ необіхдно проводити з використанням методів теорії скінченних автоматів та теорії поліномів [33, 49, 62, 63]. Відповідні методи ручного та комп'ютерного аналізу кодувальних систем були описані у підрозділах 3.8.13.3 –3.8.13.5.

3.9 Турбокоди

3.9.1 Основи теорії побудови турбокодів

Заголом турбокоди за принципом свого формування будуються на принципах побудови каскадних кодів, розглянутих у підрозділі 3.5, та згорткових кодів, розглянутих у підрозділі 3.8. Вперше ідея каскадного кодування була запропонована у 1966 р. американським математиком Д.Д. Форні, а поняття про турбокоди ввів у 1993 р. К. Берра.

Джордж Девід Форні
(народився 1940 р.)

Клод Берра
(народився 1951 р.)

В основу побудови турбокодів покладена ідея принципу максимуму правдоподібності, яка являє собою одну з загальних теоретичних основ теорії імовірності та математичної статистики та була описана у другому томі другої частини посібника [49]. Наприклад, припустимо, що двійкові логічні елементи відповідають електричним сигналам $d = +1$ та $d = -1$, де змінна є поданням біту даних, які відповідають логічному елементу. Тоді функцію правдоподібності можна записати наступним чином [33]:

$$\frac{p(x|d=+1)}{p(x|d=-1)} \underset{<H_2}{>H_1} = \frac{p(x|d=+1)P(d=+1)}{p(x|d=-1)P(d=-1)} \underset{<H_2}{>H_1} 1. \qquad (3.474)$$

В теорії кодування для формування турбокодів використовують логарифм відношення функції правдоподібності (англійський термін – Log Likelihood Ratio, LLR) (3.474) [33]:

$$L(d|x) = \lg\left(\frac{p(d=+1|x)}{p(d=-1|x)}\right) = \lg\left(\frac{p(x|d=+1)P(d=+1)}{p(x|d=-1)P(d=-1)}\right) =$$

$$= \lg\left(\frac{p(x|d=+1)}{p(x|d=-1)}\right) + \lg\left(\frac{P(d=+1)}{P(d=-1)}\right) = L(x|d) + L(d). \qquad (3.475)$$

Сутність побудови турбокодів полягає у тому, що, на відміну від згорткових кодів, в них використовуються кодери та декодери із м'якою схемою кодування та декодування. Схема кодування працює наступним чином. Під час першої ітерації всі дані на виході декодеру вважаються рівноімовірними, що, згідно з рівнянням (3.475), дає значення $L(d) = 0$.

Канальне значення LLR є результатом обчислення логарифму відношення величин l_1 та l_2 для визначених значень змінної x. Так визначається перша складова співвідношення (3.475). Вихід декодеру є результатом сумування передбаченого апостеріорного значення $L'(\hat{d})$ та зовнішнього виходу функціонального перетворювача LLR $L_e(\hat{d})$ [33]. Тобто, рівняння (3.475) можна переписати наступним чином [33]:

$$L(\hat{d}) = L_c(x) + L(d) + L_e(\hat{d}). \tag{3.476}$$

Для реалізації алгоритму роботи кодеру та декодеру турбокодів у теорії кодування вводиться поняття про алгебру функції правдоподібності [33]. Для двох статистично незалежних подій d_1 та d_2 сума логарифмів визначається наступним чином [33]:

$$L(d_1) \boxplus L(d_2) = L(d_1 \oplus d_2) = \ln\left(\frac{\exp(L(d_1)) + \exp(L(d_2))}{1 + \exp(L(d_1))\exp(L(d_2))}\right) \approx$$

$$\approx (-1) \cdot \left(\text{sign}(L(d_1)) \cdot \text{sign}(L(d_2))\right) \cdot \min(|L(d_1)|, |L(d_2)|). \tag{3.477}$$

де знак $\boxplus$ означає суму логарифмів функції правдоподібності $L(d_1 \oplus d_2)$.

3.9.2 Алгоритм ітеративного декодування турбокодів

Розглянемо алгоритм декодування турбокодів, оснований на використанні співвідношення (3.476), на прикладі ітеративного коду, який складає з k_1 рядків та k_2 стовпчиків. Узагальнений спосіб формування ітеративних кодів був розглянутий у підрозділі 3.7. Позначимо область даних, які кодуються, символом d, стовпчик для контролю парності за рядками – символом p_h, а рядок для контролю парності за стовпчиками – символом p_v. Відмінність турбокоду від стандартного ітеративного коду полягає в тому, що, крім контрольних символів p_h та p_v, до коду вводяться символи LLR, які розраховуються за співвідношеннями (3.476). Така зміна у структурі ітеративного коду надає можливість виправлення помилок. Відповідна структура турбокоду показана на рис. 3.111.

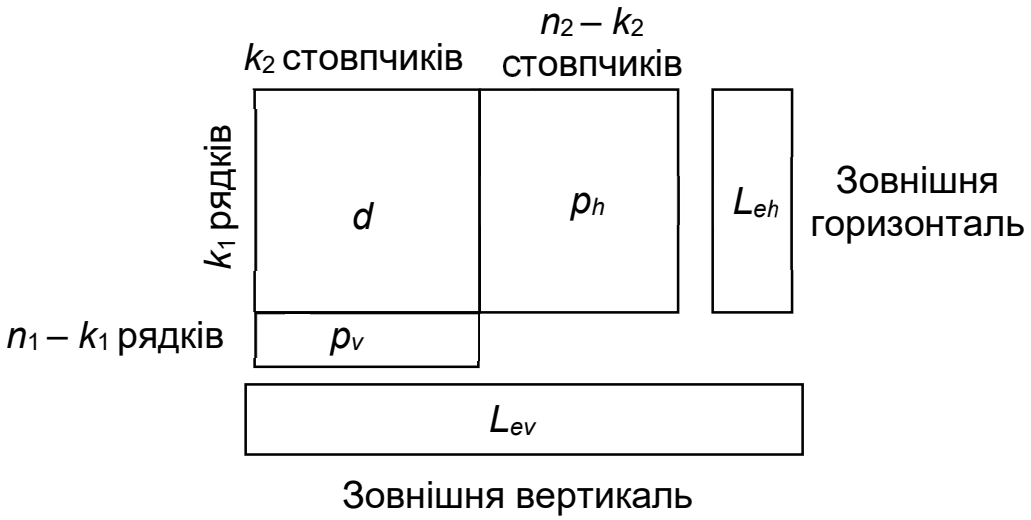


Рис. 3.111 Структура турбокоду

Для формування алгоритму розшифрування турбокоду використаємо рівняння (3.476). На основі цього рівняння формуються суми, які дають остаточне значення LLR за горизонтальним та вертикальним напрямками, а також остаточне значення $L(\hat{d})$. Для ітеративного коду, структура якого наведена на рис. 3.111, алгоритм декодування можна записати наступним чином [33].

1. Встановлюється початкове апріорне значення $L(d) = 0$.

2. Проводиться декодування горизонтального коду та, з використанням співвідношення (3.476), обчислюється значення горизонтального LLR, тобто [33]:

$$L_{eh}(\hat{d}) = L(\hat{d}) - L_c(x) - L(d). \quad (3.478)$$

3. Для наступного етапу декодування за горизонтальним напрямком встановлюється значення $L(d) = L_{eh}(\hat{d})$, де $L_{eh}(\hat{d})$ – значення, обчислене з використанням співвідношення (3.478).

4. Проводиться декодування вертикального коду та, з використанням співвідношення (3.476), обчислюється значення вертикального LLR, тобто [33]:

$$L_{ev}(\hat{d}) = L(\hat{d}) - L_c(x) - L(d). \quad (3.479)$$

5. Встановлення за результатами проведених розрахунків нового значення $L(d) = L_{ev}(\hat{d})$.

6. Якщо кількість ітерацій є достатньою – повернення до пункту 2 алгоритму із новим встановленим значенням $L(d)$, у противному випадку – перехід до пункту 7.

7. Формування остаточного рішення на виході декодеру згідно зі співвідношенням [33]:

$$L(\hat{d}) = L_c(x) + L_{ev}(\hat{d}) + L_{eh}(\hat{d}). \quad (3.480)$$

8. Закінчення роботи на виведення результату декодування.

Блок-схема описаного вище алгоритму декодування турбокоду наведена на рис. 3.112.

формуються біти даних d_i та d_j, а також біти контроля парності p_{ij}, які обчислюються через сумування значень d_i та d_j наступним за модулем два, тобто [33]:

$$d_i \oplus d_j = p_{ij}; i \in 1 \ldots k_1, j \in 1 \ldots k_2. \quad (3.481)$$

На приймальній стороні замість символів d та p приймаються символи x, для яких виконуються співвідношення [33]:

$$x_i = d_i + n; \quad x_{ij} = p_{ij} + n. \quad (3.482)$$

де n – розподіл завади, яка вважається статистично незалежною від інформаційних даних d_i та контрольних символів p_{ij}.

З урахуванням співвідношень (3.481), (3.482), співвідношення (3.475) для функції LLR у гауссовському каналі зв'язку з завадами можна переписати наступним чином [33]:

$$L_c(x_k) = \lg\left(\frac{p(x_k|d_k = +1)}{p(x_k|d_k = -1)}\right) = \ln\left(\frac{\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{x_k - 1}{\sigma}\right)^2\right)}{\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{x_k + 1}{\sigma}\right)^2\right)}\right) =$$

$$= \frac{1}{2}\left(\left(\frac{x_k+1}{\sigma}\right)^2 - \left(\frac{x_k-1}{\sigma}\right)^2\right) = \frac{2x_k}{\sigma^2}. \quad (3.483)$$

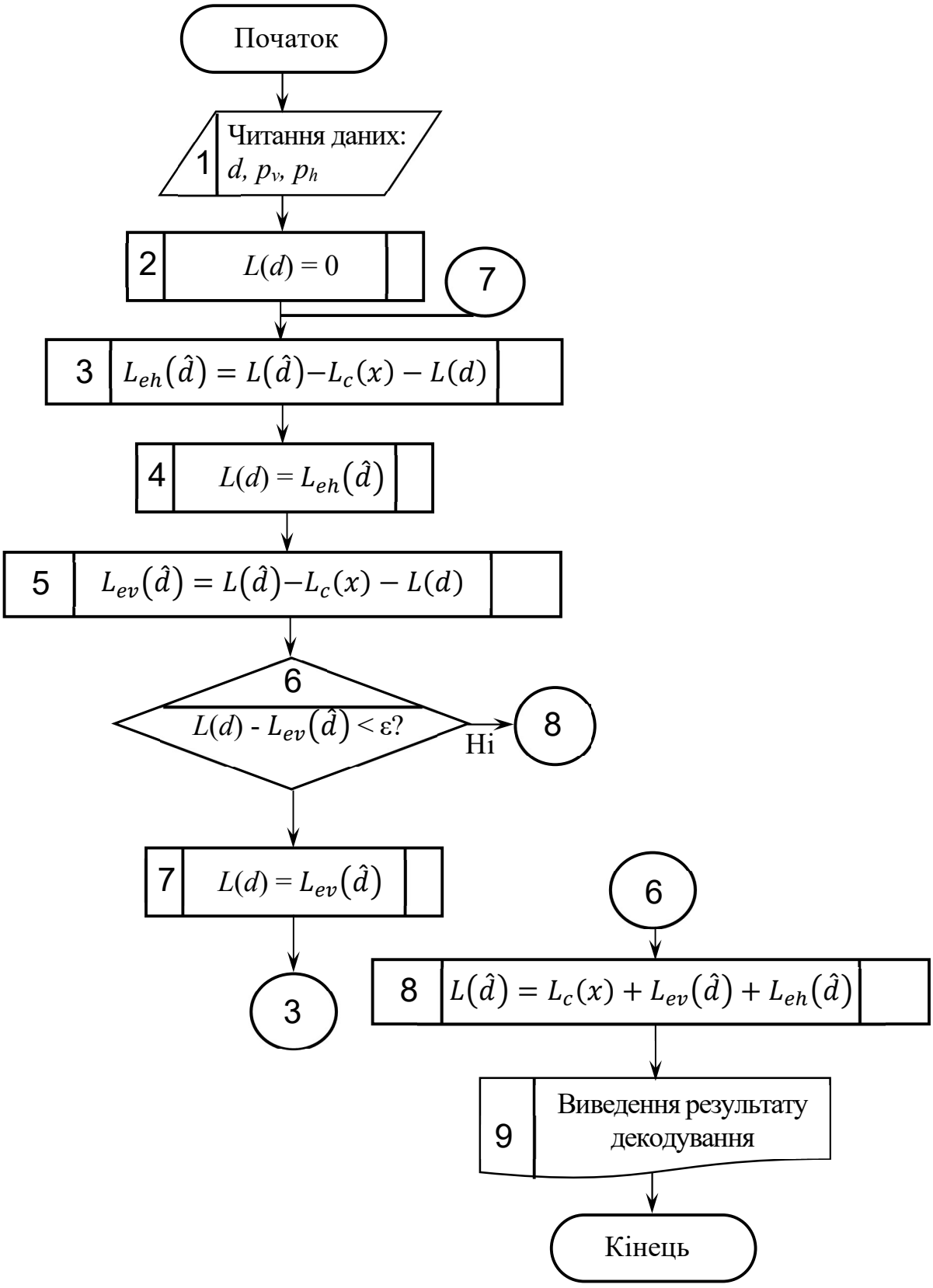


Рис. 3.112 Блок-схема алгоритму декодування ітеративного турбокоду з використанням співвідношень (3.478) – (3.480)

У ітеративному турбокоді, структура якого показана на рис. 3.111,

За умови нормованої дисперсії у гауссовському каналі зв'язку $\sigma^2 = 1$ рівняння (3.483) значно спрощується, тоді остаточний результат для функції LLR можна записати у спрощеному вигляді [33]:

$$L_c(x_k) = 2x_k. \quad (3.484)$$

Для реалізації алгоритму розшифрування ітеративного турбокоду, блок-схема якого наведена на рис. 3.112, використовуються ітераційні співвідношення (3.476), (3.477). Для першого блоку даних d_1 співвідношення (3.476) можна переписати у вигляді [33]:

$$L(\hat{d}_1) = L_c(x_1) + L(d_1) + \left(\left(L_c(x_2) + L(d_2)\right) \boxplus L_c(x_{12})\right). \quad (3.485)$$

У співвідношенні (3.485) складова $\left(\left(L_c(x_2) + L(d_2)\right) \boxplus L_c(x_{12})\right)$ являє собою зовнішні функції LLR, задані співвідношеннями (3.483) та (3.484), які безпосередньо генеруються в процесі розшифрування послідовності турбокоду [33].

Аналогічно, для будь-якого блоку даних d_i можна записати [33]:

$$L(\hat{d}_i) = L_c(x_i) + L(d_i) + \left(\left(L_c(x_j) + L(d_j)\right) \boxplus L_c(x_{ij})\right), \quad (3.486)$$

де $L_c(x_i)$, $L_c(x_j)$, $L_c(x_{ij})$ – канальні вимірювання приймачем функцій LLR для відповідних даних d_i, d_j та p_{ij}, $L(d_i)$, $L(d_j)$ – апріорні функції LLR для тих самих даних d_i, d_j.

Розглянемо описаний у цьому підрозділі спосіб декодування ітеративного турбокоду на конкретному прикладі.

3.9.3 Приклад формування та розшифрування ітеративного турбокоду

Перед вивченням цього підрозділу необхідно повторити розділ 1.

Розглянемо тепер приклад формування та розшифрування ітеративного турбокоду, структура якого показана на рис. 3.111. Спосіб формування та

розшифрування турбокоду, який буде розглядатися, оснований на використанні ітераційного алгоритму, блок-схема якого наведена на рис. 3.112, та співвідношень (3.481) – (3.486).

Приклад 3.71 Дані, які передаються з використанням турбокоду, мають наступну структуру: $d_1 = 1$, $d_2 = 0$, $d_3 = 0$, $d_4 = 1$. З використанням цих даних побудувати на основі співвідношень (3.481) – (3.486) відповідний ітеративний турбокод, занести в нього помилку та показати, що ця помилка виявляється та виправляється на етапі декодування [33].

Спочатку, з використанням співвідношення (3.481), знайдемо значення контрольних символів p_{12}, p_{13}, p_{24} та p_{34}. Відповідно, маємо:

$$p_{12} = d_1 \oplus d_2 = 1 \oplus 0 = 1; \qquad p_{13} = d_1 \oplus d_3 = 1 \oplus 0 = 1;$$

$$p_{24} = d_2 \oplus d_4 = 0 \oplus 1 = 1; \qquad p_{34} = d_3 \oplus d_4 = 0 \oplus 1 = 1.$$

Тоді кодова послідовність для сформованого турбокоду, яка передається через канал зв'язку, буде мати наступний вигляд:

$$\mathbf{T} = \{\{d_i\}, \{p_{ij}\}\} = \{1, 0, 0, 1, 1, 1, 1, 1\}. \qquad (3.487)$$

Припустимо, що інформаційні біти подаються в каналі зв'язку біполярним кодом AMI із значеннями напруги +1 В, яке відповідає логічному рівню 1, та –1 В, що відповідає логічному рівню 0. Способи кодування електричних сингалів у каналах зв'язку були досконало розглянуті у першому розділі цієї частини посібника. За такої умови послідовність **T**, задана співвідношенням (3.487), перепишеться наступним чином:

$$\mathbf{T} = \{\{d_i\}, \{p_{ij}\}\} = \{+1, -1, -1, +1, +1, +1, +1, +1\}. \qquad (3.488)$$

Припустимо, що через наявність завади в каналі зв'язку вектор **T**, заданий співвідношенням (3.488), був спотворений наступним чином:

$$\mathbf{T_c} = \{\{x_i\}, \{x_{ij}\}\} = \{0{,}75, 0{,}05, 0{,}10, 0{,}15, 1{,}25, 1{,}0, 3{,}0, 0{,}5\}. \qquad (3.489)$$

Для обчислення функції правдоподібності $L_c(x_i)$ скористаємося співвідношенням (3.484). Відповідно, маємо:

$$\{\{L_c(x_i)\}, \{L_c(x_{ij})\}\} = \{1{,}5, 0{,}1, 0{,}2, 0{,}3, 2{,}5, 2{,}0, 6{,}0, 1\} =$$

$$= \{x_1, x_2, x_3, x_4, x_{12}, x_{34}, x_{13}, x_{24}\}. \quad (3.490)$$

Значення, задані співвідношенням (3.490), являють собою результати вимірювань, які надходять на вхід декодеру. Слід відзначити, що за визначених умов обчислені значення функції правдоподібності LLR дають хибний результат для значень x_2 та x_3. Декодер на цьому етапі розшифрування коду, згідно зі співвідношенням (3.490), прийме рішення, що $x_2 = 1$ та $x_3 = 1$, а за умовою завдання $d_1 = 0$ та $d_2 = 0$. Зрозуміло, що причиною цієї помилки декодування є наявність завади в каналі зв'язку.

З використанням ітераційного співвідношення (3.486), а також співвідношень (3.478), (3.479), записаних для горизонтального та вертикального значень LLR, будемо шукати інші значення функції правдоподібності LLR на наступних ітераціях. Відповідно, на другій ітерації, для значень горизонтальної функції LLR, такі обчислення мають наступний вигляд:

$$L_{eh}(\hat{d}_1) = \left(L_c(x_2) + L(d_2)\right) \boxplus L_c(x_{12}) = (0{,}1 + 0) \boxplus 2{,}5 = -0{,}1;$$
$$L_{eh}(\hat{d}_2) = \left(L_c(x_1) + L(d_1)\right) \boxplus L_c(x_{12}) = (1{,}5 + 0) \boxplus 2{,}5 = -1{,}5;$$
$$L_{eh}(\hat{d}_3) = \left(L_c(x_4) + L(d_4)\right) \boxplus L_c(x_{34}) = (0{,}3 + 0) \boxplus 2 = -0{,}3; \quad (3.491)$$
$$L_{eh}(\hat{d}_4) = \left(L_c(x_3) + L(d_3)\right) \boxplus L_c(x_{34}) = (0{,}2 + 0) \boxplus 2 = -0{,}2.$$

У співвідношеннях (3.491) всі значення $L_{eh}(\hat{d}_1)$, $L_{eh}(\hat{d}_2)$, $L_{eh}(\hat{d}_3)$ та $L_{eh}(\hat{d}_4)$ відповідають новому значенню $L(d_1)$.

Тепер слід обчислити значення вертикальної функції LLR, але з урахуванням вже отриманих значень $L(d_1) - L(d_4)$ для горизонтальної функції, які задані співвідношеннями (3.491). Відповідно, можна записати:

$$L_{ev}(\hat{d}_1) = \left(L_c(x_3) + L(d_3)\right) \boxplus L_c(x_{13}) = (0{,}2 - 0{,}3) \boxplus 6 = 0{,}1;$$
$$L_{ev}(\hat{d}_2) = \left(L_c(x_1) + L(d_1)\right) \boxplus L_c(x_{12}) = (1{,}5 + 0) \boxplus 2{,}5 = -1{,}5;$$
$$L_{ev}(\hat{d}_3) = \left(L_c(x_1) + L(d_1)\right) \boxplus L_c(x_{13}) = (1{,}5 + 0) \boxplus 2{,}5 = -1{,}5; \quad (3.492)$$
$$L_{ev}(\hat{d}_4) = \left(L_c(x_2) + L(d_2)\right) \boxplus L_c(x_{24}) = (0{,}1 - 1{,}5) \boxplus 1{,}0 = 1{,}0.$$

У співвідношеннях (3.492) $L_{ev}(\hat{d}_1)$ відповідає новому значенню $L(d_1)$, $L_{ev}(\hat{d}_2)$ – новому значенню $L(d_2)$, $L_{ev}(\hat{d}_3)$ – новому значенню $L(d_3)$, а $L_{ev}(\hat{d}_4)$ –

новому значенню $L(d_4)$. Відповідно, після горизонтального декодування $L_{eh}(\hat{d}_1) = 1{,}4;\ L_{eh}(\hat{d}_2) = -1{,}4;\ L_{eh}(\hat{d}_3) = -0{,}1;\ L_{eh}(\hat{d}_4) = 0{,}1.$ Відповідно, сумування за результатами горизонтального та вертикального кодування дає наступний результат: $L(d_1) = 1{,}5$; $L(d_2) = -1{,}5$; $L(d_3) = -1{,}5$; $L(d_4) = 1$.

З результатів використання алгоритму розшифрування турбокоду, основаних на блок-схемі, наведеній на рис. 3.112 та заданих співвідношеннями (3.491), (3.492), зрозуміло, що такий алгоритм значно підвищує рівень функції правдоподібності L для інформаційних даних d_1, d_2, d_3 та d_4. Подальші розрахунки дають наступний результат:

$$\begin{gathered} L_{eh}(\hat{d}_1) = (0{,}1 - 0{,}1) \boxplus 2{,}5 = 0; \\ L_{eh}(\hat{d}_2) = (1{,}5 - 0{,}1) \boxplus 2{,}5 = -1{,}6; \\ L_{eh}(\hat{d}_3) = (0{,}3 - 0{,}1) \boxplus 2 = -1{,}3; \\ L_{eh}(\hat{d}_4) = (0{,}2 - 1{,}4) \boxplus 2 = 1{,}2. \end{gathered} \tag{3.493}$$

Відповідно, для вертикального перетворення маємо:

$$\begin{gathered} L_{ev}(\hat{d}_1) = (0{,}2 - 1{,}3) \boxplus 6 = 1{,}1; \\ L_{ev}(\hat{d}_2) = (0{,}3 + 1{,}2) \boxplus 1{,}0 = -1{,}0; \\ L_{ev}(\hat{d}_3) = (1{,}5 + 0) \boxplus 6 = -1{,}5; \\ L_{ev}(\hat{d}_4) = (0{,}1 - 1{,}6) \boxplus 1{,}0 = 1{,}0. \end{gathered} \tag{3.494}$$

Згідно зі співвідношеннями (3.493), після горизонтального декодування маємо наступні результати: $L(d_1) = 0$; $L(d_2) = -1{,}6$; $L(d_3) = -1{,}3$; $L(d_4) = 1{,}2$.

Відповідно, після вертикального декодування, згідно зі співвідношеннями (3.494): $L(d_1) = 1{,}1$; $L(d_2) = -1{,}0$; $L(d_3) = -1{,}5$; $L(d_4) = 1{,}0$.

В результаті, після другої ітерації, згідно зі співвідношенням (3.480) отримуємо наступний результат: $L(d_1) = 2{,}6$; $L(d_2) = -2{,}5$; $L(d_3) = -2{,}6$; $L(d_4) = 2{,}5$.

Цей результат є правдоподібним. Дійсно, згідно з отриманими значеннями функції правдоподібності $d_1 = 1$, $d_2 = 0$, $d_3 = 0$, $d_4 = 1$, що відповідає умові задачі. Тобто, задачу можна вважати розв’язаною.

3.9.4 Формування турбокодів на основі згорткових кодів

Розглянутий у підрозділах 3.9.2 та 3.9.3 ітеративний турбокод є найпростішим варіантом такого коду, але він не оптимальним з точки зору коректувальних можливостей. Більш продуктивною є ідея реалізації турбокодів через паралельне сполучення не лінійних, а згорткових кодів [33].

Розглянемо прості двійкові згорткові коди з довжиною буферу K та степінню надлишковості $\frac{1}{n} = \frac{1}{2}$. Кодер турбокоду, побудованого на основі згорткового коду, працює дуже просто. На вхід кодеру у момент часу k подаватимемо біт d_k, а відповідну пару бітів на виході кодеру будемо формувати наступним чином [33]:

$$u_k = \sum_{i=0}^{K-1} g_{1i} d_{k-i}\,, v_k = \sum_{i=0}^{K-1} g_{2i} d_{k-i}, \qquad (3.495)$$

де $\mathbf{G}_1 = \{g_{1i}\}$ та $\mathbf{G}_2 = \{g_{2i}\}$ – два генератори згорткового коду.

Структурна схема кодеру несистематичного згорткового коду, робота якого базується на використанні системи рівнянь (3.495), наведена на рис. 3.113.

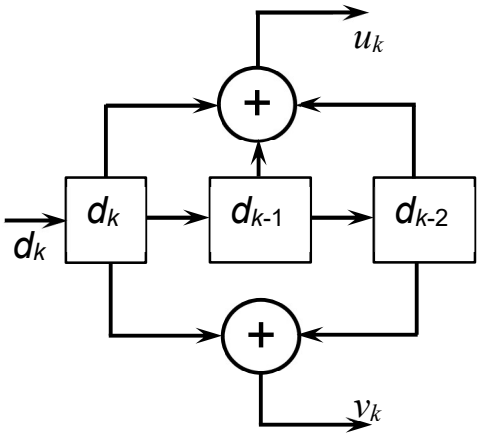


Рис. 3.113 Кодер турбокоду, побудованого на основі несистематичних згорткових кодів

Для кодеру турбокоду, структурна схема якого наведена на рис. 3.113, довжина кодового обмеження складає $K = 3$, а для генерації згорткових кодів використані кодові послідовності $\mathbf{G}_1 = \{111\}$ та $\mathbf{G}_2 = \{101\}$. Подібні схеми

кодування на основі несистематичних згорткових кодів вельми добре працюють за умови великих значень співвідношення потужності сигналу до потужності шуму $\left(\frac{E_b}{N_0}\right)$ [33]. У разі малих значень відношення потужності сигналу до потужності шуму $\left(\frac{E_b}{N_0}\right)$ більш ефективним є інший підхід, коли як складові компоненти турбокодів використовуються згорткові коди з нескінченною імпульсною характеристикою [33]. Такі самі компоненти також використовуються в рекурсивних несистематичних згорткових кодах, відмінність яких полягає в тому, що в них попередньо закодовані біти даних постійно знову надходять на вхід кодеру. За умови високої степені кодування турбокоди, побудовані на основі систематичних згорткових кодів, дають значно кращі результати, ніж турбокоди, побудовані на основі несистематичних кодів [33]. Структурна схема кодеру, який формує турбокод на основі систематичних згорткових кодів, наведена на рис. 3.114. Відмінними рисами цієї схеми є наявність контуру зворотного зв'язку, а також те, що один з двох виходів, u_k або v_k, встановлюється рівним значенню вхідного біту d_k. Для схеми, наведеної на рис. 3.114, довжина кодового обмеження складає $K = 3$, а значення a_k є результатом виконання рекурсивної процедури [33]:

$$a_k = d_k + \sum_{i=0}^{K-1} g_i' a_k, \qquad g_i' = \begin{cases} g_{1i}, & \text{якщо } u_k = d_k; \\ g_{2i}, & \text{якщо } v_k = d_k. \end{cases} \tag{3.496}$$

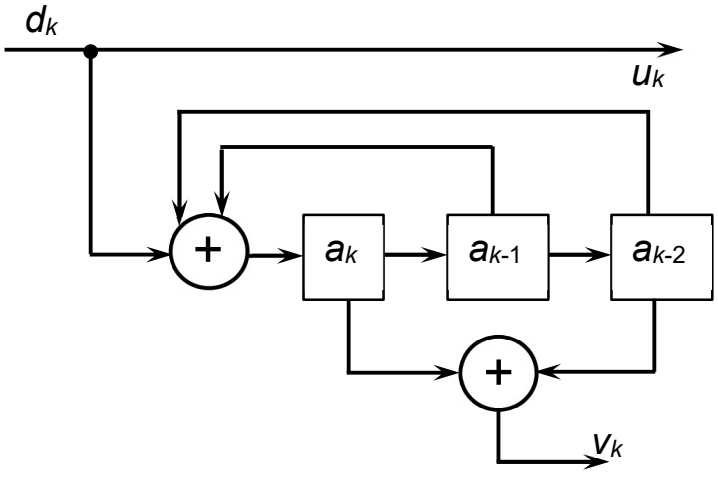


Рис. 3.114 Кодер турбокоду, побудованого на основі систематичних згорткових кодів

Зазвичай вважається, що для кодерів, наведених на рис. 3.113 та 3.114 вхідний біт d_k приймає значення 0 та 1 із однаковою імовірністю. Крім цього, можна обґрунтувати, що послідовності $\{a_k\}$ та $\{d_k\}$ мають однакові статистичні імовірності [33]. Параметр просвіту для турбокодів, сформованих на основі систематичних та несистематичних згорткових кодів, також є однаковим [33]. Більш досконалі відомості про різні типи турбокодів та їхні параметри також наведені у навчальному посібнику [33].

3.10 Порівняльний аналіз завадостійких кодів та перспективи їхнього подальшого розвитку

У попередніх підрозділах цього розділу розглядалися способи формування завадостійких кодів, які сьогодні використовуються в системах зв'язку, в комп'ютерних та в електронних системах. Проте сама теорія кодування цифрової інформації постійно розвивається, і, за рахунок цього, розробляються та знаходять практичне впровадження нові методи та способи кодування. Математичним підґрунтям для розвитку нових методів кодування є теорія груп, теорія кінцевих полів, теорія матриць та теорія скінченних автоматів [48].

Загалом у сучасних телекомунікаційних системах використовують різні модифікації циклічних та згорткових кодів. Насамперед, це коди БЧХ, які розглядалися у підрозділі 3.3.4. Загалом вони є аналогом циклічних кодів та відрізняються способами формування утворювальних поліномів. Проте більш ефективними з точки зору стиснення інформації та можливостей виправлення помилок є багатопозиційні циклічні коди. Широко використовуються у сучасних системах зв'язку розглянуті у підрозділі 3.4 коди Ріда – Соломона. Зокрема РС-коди знаходять широке впровадження в комп'ютерних системах та в системах обробки звукової та відеоінформації за умови відсутності потужних завад. Наприклад, з використанням РС-кодів в комп'ютерних системах здійснюється записування інформації на магнітні та оптичні носії. Способи формування та розшифровування РС-кодів розглядалися у підрозділі 3.4.

Розробляються та знаходять впровадження у сучасних системах зв’язку різні типи ітеративних кодів. Це дуже широкий клас кодів, які базуються на засобах циклічного кодування, але з елементами теорії матриць та комбінаторних методів. Загальні теоретичні основи побудови ітеративних кодів розглядалися у підрозділі 3.7. Формування ітеративного коду основане на ітераційних обчисленнях, проте, на відміну від згорткових кодів, використаються не векторні, а матричні ітераційні алгоритми. Під час формування матриці ітеративних кодів застосовують два кодових слова: зовнішній код, який записується у рядках, та внутрішній, який записується у стовпчиках. Но основі теорії ітеративних кодів побудовані також турбокоди, які розглядалися в підрозділі 3.9. Відмінною рисою цього типу кодування є те, що використання матриць ітеративних кодів дозволяє зменшити обчислювальну складність алгоритмів кодування та поліпшити коректувальні параметри відносно кодів БЧХ та РС.

Однією із поширених модифікацій ітеративних кодів є каскадні коди, які запропонував Форні. Вони розглядалися в підрозділі 3.5. Головна відмінність каскадних кодів від ітеративних полягає у тому, що внутрішній код формується на основі зовнішнього, що сприяє подальшому спрощенню ітераційного алгоритму.

На основі матричних методів базується ще один вкрай поширений клас кодів низької щільності, які були запропоновані Р. Галлагером. Головною ідеєю алгоритмів кодування кодів низької щільності є використання розряджених перевірочних матриць, що призводить до суттєвого зменшення їх обчислювальної складності [28, 29].

Проте найпростішими з точки зору апаратної реалізації кодерів та декодерів є згорткові коди, і тому саме вони найбільш часто використовуються у різноманітних стандартах сучасних систем проводового та безпроводового зв’язку. Загальна ідея побудови згорткових кодів досить проста та була розглянута у підрозділі 3.8.

Одним із останніх досягнень математичної теорії кодування є

турбокоди, розглянуті в підрозділі 3.9. Основною їх ідеєю є суміщення алгоритмів ітеративних кодів та багаторозрядного завадостійкого кодування. Як і звичайні рівномірні завадостійкі коди, турбокоди поділяються на класи блокових та згорткових кодів.

Останнім часом розвивається теорія узагальнених каскадних кодів, запропонована групою вчених: Е. Блохом, В. Зябловим та В. Зіновієвим. Ця теорія основана на використанні двійкових векторів та на теорії груп.

Засоби кодування, які знайшли впровадження у стандартах сучасних безпроводових комп'ютерних мереж та систем зв'язку, систематизовані у таблиці 3.33.

Таблиця 3.33 – Способи кодування сигналів, які використовуються в сучасних стандартах безпроводового зв'язку та в комп'ютерних мережах [29]

Стандарт	Методи кодування
IEEE 802.3 a n	Код низької щільності, оснований на коді РС
IEEE 802.11	Згортковий код Вітербі
IEEE 802.16	Перший етап кодування – простий код з перевіркою на парність та коди Хеммінга. Другий етап кодування – код Вітербі та згорткові турбоходи.
3GGP LTE	Комбінація загорткових кодів Вітербі та турбокодів.

Контрольні питання та завдання до розділу 3

1. Сформулюйте загальні положення та принципи, які використовуються для формування групових завадостійких кодів. Чим групові коди суттєво відрізняються від лінійних та циклічних кодів? Свою відповідь обґрунтуйте на наведіть доречні приклади.

2. Поясніть співвідношення (3.1) – (3.15) та наведіть приклади їх використання для формування групових кодів.

3. Поясніть співвідношення (3.16) – (3.18) та спосіб формування кодів

Файра.

4. Поясніть структурні схеми кодувальних та декодувальних цифрових електронних пристроїв, призначених для формування та декодування послідовностей кодів Файра, які наведені на рис. 3.1 та рис. 3.2.

5. Поясніть приклади 3.1 та 3.2.

6. Поясніть блок-схему алгоритму, яка наведена на рис. 3.3.

7. Побудувати код Файра для заданої вхідної послідовності C із заданими коректувальними параметрами m та b.

а) $C = [1,0,1,1,0,1,1]$, $m = 5$, $b = 2$; б) $C = [1,1,1,1,0,1,1]$, $m = 5$, $b = 3$;

в) $C = [1,0,1,1,0,0,1]$, $m = 7$, $b = 3$; г) $C = [1,1,1,1,0,1,1]$, $m = 7$, $b = 2$;

д) $C = [1,0,1,1,1,0,1]$, $m = 7$, $b = 3$; е) $C = [1,1,0,1,0,1,1]$, $m = 5$, $b = 2$;

є) $C = [1,0,1,0,1,0,1]$, $m = 5$, $b = 3$; ж) $C = [1,1,0,1,0,1,1]$, $m = 7$, $b = 2$;

з) $C = [1,0,1,1,1,0,1]$, $m = 5$, $b = 2$; і) $C = [1,1,1,1,0,0,1]$, $m = 7$, $b = 3$;

к) $C = [1,0,0,1,0,0,1]$, $m = 5$, $b = 3$; л) $C = [1,1,0,1,0,0,1]$, $m = 7$, $b = 2$.

За завданням викладача занести у розряди сформованого коду подвійну та потрійну помилку та показати, як її можна виправити.

8. Поясніть код програми, яка наведена у додатку Л, та результати виконання цієї програми.

9. Що являють собою коди БЧХ та який головний принцип їх побудови? Наведіть власний приклад формування коду БЧХ.

10. Які галузі застосування кодів БЧХ у сучасній електронній апаратурі та у системах зв’язку Вам відомі?

11. Що являє собою конструктивна відстань кодів БЧХ та як цей параметр коду пов’язаний із мінімальною кодовою відстанню та максимальною кількістю помилок, які виправляються?

12. Поясніть визначення 3.1 та формулу (3.22). Наведіть власні приклади обчислення конструктивної кодової відстані для коду БЧХ.

13. Які головні параметри кодів БЧХ та як вони пов’язані між собою?

Поясніть співвідношення (3.21) – (3.24) та теореми 3.1 – 3.3.

14. Опишіть у загальній формі алгоритм формування кодів БЧХ.

15. Поясніть співвідношення (3.25) та блок-схему алгоритму, наведену на рис. 3.5.

16. Поясніть числові дані, які наведені у таблицях 3.1 та 3.2.

17. Поясніть приклади 3.3 – 3.12. Наведіть власні приклади формування коду БЧХ та декодування сформованої кодової послідовності.

18. Які способи декодування кодів БЧХ Вам відомі? Поясніть узагальнену схему декодера коду БЧХ, яка наведена на рис. 3.6.

19. У чому полягає сутність алгоритму Пітерсона – Горенстейна – Цирлера (ПГЦ) та як він використовується для декодування кодів БЧХ?

20. Що являє собою матриця Вандермонда та яким чином вона пов’язана із алгоритмами декодування кодів БЧХ?

21. Чи може бути використана матриця Вандермонда для декодування кодів Ріда – Соломона? Свою відповідь обґрунтуйте.

22. Що являє собою процедура Ченя та яким чином вона пов’язана із алгоритмами декодування кодів БЧХ?

23. Чи може бути використана процедура Ченя для декодування кодів Ріда – Соломона? Свою відповідь обґрунтуйте.

24. Поясніть співвідношення (3.37) – (3.70).

25. Поясніть блок-схему алгоритму, наведену на рис. 3.7.

26. Поясніть теореми 3.4 та 3.5.

27. Поясніть приклад 3.13.

28. У чому полягає сутність алгоритму Берлекемпа – Мессі та як він використовується для декодування кодів БЧХ?

29. Чому алгоритм Берлекемпа – Мессі вважається більш ефективним з обчислювальної точки зору, ніж алгоритм ПГЦ?

30. Що являє собою процедура Форні та яким чином вона пов’язана із

алгоритмами декодування кодів БЧХ?

31. Поясніть співвідношення (3.79) – (3.83) та теорему 3.6.

32. Поясніть співвідношення (3.84) та теорему 3.7.

33. Поясніть співвідношення (3.85) – (3.91).

34. Поясніть блок-схему алгоритму, наведену на рис. 3.8.

35. Поясніть приклад 3.14.

36. Поясніть числові дані, які наведені у таблиці 3.3.

37. Поясніть співвідношення (3.92) та (3.93).

38. У чому полягає сутність алгоритму Евкліда для поліномів та як він може бути використаний для декодування кодів БЧХ?

39. Поясніть сутність теорем 3.8 та 3.9.

40. Поясніть співвідношення (3.94) – (3.118).

41. Поясніть приклади 3.15 та 3.16.

42. Поясніть блок-схему алгоритму, наведену на рис. 3.13.

43. Поясніть узагальнені структурні схеми декодерів кодів БЧХ, які наведені на рис. 3.14 – 3.16.

44. Поясніть спосіб формування систематичного коду БЧХ, який виявляє та виправляє подвійні помилки.

45. Поясніть співвідношення (3.120) – (3.127).

46. Поясніть співвідношення (3.128) – (3.132) та числові дані, які наведені у таблиці 3.4.

47. Поясніть принципову електричну схему цифрового кодувального пристрою, наведену на рис. 3.17.

48. Поясніть співвідношення (3.133) – (3.139) та числові дані, які наведені у таблиці 3.5.

49. Поясніть блок-схему алгоритму, наведену на рис. 3.18.

50. Поясніть приклад 3.17.

51. Поясніть принципову електричну схему цифрового декодувального

пристрою, наведену на рис. 3.19.

52. Поясніть приклад 3.18.

53. Поясніть числові дані, які наведені у таблиці 3.7.

54. Поясніть співвідношення (3.149) – (3.151).

55. Поясніть співвідношення (3.152).

56. Поясніть спосіб формування систематичного коду БЧХ, який виявляє та виправляє потрійні помилки.

57. Поясніть принципову електричну схему цифрового кодувального пристрою, наведену на рис. 3.20.

58. Поясніть числові дані, які наведені у таблицях 3.8 та 3.9.

59. Поясніть співвідношення (3.153) – (3.157) та числові дані, які наведені у таблиці 3.10.

60. Поясніть приклад 3.19 та співвідношення (3.158) – (3.197).

61. Поясніть блок-схему алгоритму, наведену на рис. 3.21.

62. Поясніть принципову електричну схему цифрового кодувального пристрою, наведену на рис. 3.22.

63. Поясніть принципову електричну схему цифрового декодувального пристрою, наведену на рис. 3.23.

64. Чи можуть бути використані для декодування кодів БЧХ алгоритми декодування циклічних кодів, які розглядалися у підрозділі 2.5? Свою відповідь обґрунтуйте та наведіть власні приклади такого використання алгоритмів декодування циклічних кодів.

65. Поясніть приклади 3.20, 3.21, 3.22 та рис. 3.24 – 3.27.

66. Поясніть код програми **BCH**, наведеної у додатку М.

67. Поясніть блок-схему алгоритму, наведену на рис. 3.28.

68. Для заданої вхідної послідовності **C** побудувати код БЧХ (15, 7). Занести дві помилки у визначені розряди кодової послідовності n_1 та n_2 та показати, яким чином ці помилки можна виправити.

а) $\mathbf{C} = [1,0,1,1,0,1,1]$, $n_1 = 3, n_2 = 10$; б) $\mathbf{C} = [1,0,1,1,0,0,1]$, $n_1 = 5, n_2 = 7$;

в) $\mathbf{C} = [1,0,1,0,0,1,1]$, $n_1 = 8, n_2 = 10$; г) $\mathbf{C} = [1,1,1,1,0,1,1]$, $n_1 = 9, n_2 = 10$;

д) $\mathbf{C} = [1,1,1,0,0,1,1]$, $n_1 = 2, n_2 = 5$; е) $\mathbf{C} = [1,1,0,0,0,1,1]$, $n_1 = 4, n_2 = 11$;

є) $\mathbf{C} = [1,0,0,0,1,1,0]$, $n_1 = 6, n_2 = 12$; ж) $\mathbf{C} = [1,1,0,1,1,1,1]$, $n_1 = 4, n_2 = 7$;

з) $\mathbf{C} = [1,0,1,0,0,1,0]$, $n_1 = 5, n_2 = 10$; і) $\mathbf{C} = [1,1,0,1,0,1,0]$, $n_1 = 6, n_2 = 13$;

к) $\mathbf{C} = [1,1,1,0,1,1,1]$, $n_1 = 5, n_2 = 12$; л) $\mathbf{C} = [1,0,0,1,1,1,0]$, $n_1 = 8, n_2 = 14$.

Перевірити отримані результати з використанням програми **ВСН**, код якої наведений у додатку М.

69. Для заданої вхідної послідовності **С** побудувати код БЧХ (15, 5). Занести три помилки у визначені розряди кодової послідовності n_1, n_2 та n_3 та показати, яким чином ці помилки можна виправити.

а) $\mathbf{C} = [1,0,1,1,0]$, $n_1 = 3, n_2 = 5, n_3 = 9$; б) $\mathbf{C} = [1,1,1,0,0]$, $n_1 = 4, n_2 = 7, n_3 = 12$;

в) $\mathbf{C} = [1,0,1,1,1]$, $n_1 = 7, n_2 = 10, n_3 = 12$; г) $\mathbf{C} = [1,1,1,0,0]$, $n_1 = 1, n_2 = 4, n_3 = 6$;

д) $\mathbf{C} = [1,0,1,0,1]$, $n_1 = 4, n_2 = 6, n_3 = 9$; е) $\mathbf{C} = [1,1,0,1,1]$, $n_1 = 2, n_2 = 7, n_3 = 14$;

є) $\mathbf{C} = [1,1,1,0,1]$, $n_1 = 2, n_2 = 6, n_3 = 8$; ж) $\mathbf{C} = [1,1,0,1,1]$, $n_1 = 6, n_2 = 7, n_3 = 11$;

з) $\mathbf{C} = [1,0,1,0,0]$, $n_1 = 3, n_2 = 5, n_3 = 6$; і) $\mathbf{C} = [1,1,0,1,1]$, $n_1 = 2, n_2 = 3, n_3 = 8$;

к) $\mathbf{C} = [1,0,0,0,1]$, $n_1 = 5, n_2 = 8, n_3 = 9$; л) $\mathbf{C} = [1,1,1,1,1]$, $n_1 = 8, n_2 = 9, n_3 = 13$.

Перевірити отримані результати з використанням програми **ВСН**, код якої наведений у додатку М.

70. У яких сучасних пристроях цифрової електронної апаратури використовуються коди Ріда – Соломона? Наведіть приклади використання кодів Ріда – Соломона у сучасні електронній апаратурі.

71. Поясніть визначення 3.2 та співвідношення (3.199), (3.200).

72. Чому коди Ріда – Соломона зазвичай розглядаються як один із різновидів кодів БЧХ? Свою відповідь обґрунтуйте.

73. Що являє собою кодова відстань багатопозиційної кодової комбінації та як вона обчислюється? Чим відрізняється цей параметр від кодової відстані для двійкового коду? Наведіть власні приклади обчислення кодової відстані

для багатопозиційної кодової комбінації.

74. Що являє собою мінімальна кодова відстань багатопозиційної кодової комбінації та як вона обчислюється? Чим відрізняється цей параметр від мінімальної кодової відстані для двійкового коду? Наведіть власні приклади обчислення мінімальної кодової відстані для багатопозиційної кодової комбінації.

75. Поясніть теорему 3.10 та співвідношення (3.203).

76. Що являє собою процес стирання символів та як він розглядається у теорії завадостійкого кодування сигналів? Поясніть визначення 3.5.

77. Що являють собою пакетні помилки та як вони розглядаються у теорії завадостійкого кодування сигналів? Поясніть визначення 3.6.

78. Які параметри пакетів помилок Вам відомі та як вони визначаються? Поясніть визначення 3.7 та 3.8, а також співвідношення (3.204) – (3.207).

79. Що являє собою періодичність пакетів помилок та як вона обчислюється? Наведіть власні приклади обчислення періодичності пакетів помилок.

80. Що являє собою інтенсивність пакетів помилок та як вона обчислюється? Наведіть власні приклади обчислення інтенсивності пакетів помилок.

81. Що являє собою відображення багатопозиційного коду Ріда – Соломона на двійкові коди та як воно використовується у теорії кодування сигналів? Наведіть власні приклади реалізації такого відображення.

82. Поясніть приклад 3.23 та числові дані, які наведені в таблицях 3.11, 3.12.

83. Поясніть приклад 3.24.

84. Що являють собою поля Галуа та як вони використовуються для формування кодів Ріда – Соломона?

85. Як здійснюється в алгебрі полів Галуа сумування елементів поля? Наведіть власні приклади виконання цієї операції.

86. Поясніть приклади 3.25 та 3.26.

87. Як у полях Галуа $GF(2^m)$ виконується алгебраїчна операція піднесення числа 2 до відповідної степені. Поясніть співвідношення (3.209), (3.210). Наведіть власні приклади обчислення функції $f(i) = 2^i$ для різних полів Галуа.

88. Як у полях Галуа $GF(2^m)$ обчислюється логарифмічна функція? Наведіть власні приклади обчислення логарифмічної функції для різних полів Галуа.

89. Як у полях Галуа $GF(2^m)$ визначаються алгебраїчні операції множення та ділення? Поясніть визначення 3.10, 3.11, та співвідношення (3.211), (3.212).

90. Поясніть числові дані, які наведені у таблицях 3.13 – 3.15.

91. Чому поля Галуа $GF(2^m)$, які використовуються для формування кодів Ріда – Соломона, повинні відповідати властивості мультиплікативності для операції множення? Свою відповідь обґрунтуйте з точки зору теорії завадостійкого кодування та теорії полів Галуа.

92. Що являє собою нульовий елемент мультиплікативного поля Галуа і яку роль відіграє цей елемент в процесі формування завадостійких групових кодів? Свою відповідь обґрунтуйте з точки зору теорії завадостійкого кодування та теорії полів Галуа.

93. Поясніть приклад 3.27 та числові дані, які наведені у таблицях 3.16 та 3.17.

94. Що являються собою обернені елементи полів Галуа та як вони обчислюються? Поясніть співвідношення (3.213). Наведіть власні приклади обчислення обернених елементів у заданому полі Галуа.

95. Поясніть приклади 3.28, 3.29. Наведіть власні приклади множення двох елементів заданого поля Галуа.

96. Поясніть приклад 3.30. Наведіть власні приклади ділення двох

елементів заданого поля Галуа.

97. Чому у теорії автоматизації обчислень алгебраїчні операції множення та ділення елементів поля Галуа вважається дуже ресурсоємними з точки зору витрат машинного часу, та у разі написання швидкодійного програмного забезпечення для розв'язування складних завдань теорії кодування сигналів використання цих операцій зазвичай намагаються уникнути. Свою відповідь обґрунтуйте з точки зору теорії полів Галуа.

98. Чому алгебраїчні операції над полями Галуа зазвичай не потребують виконання великої кількості арифметичних перевірок, які часто використовуються у комп'ютерній арифметиці для реалізації коректних дій із дійсними та раціональними числами? Свою відповідь обґрунтуйте з точки зору теорії полів Галуа.

99. Чому зазвичай у сучасних комп'ютерних системах, призначених для проведення науково-технічних розрахунків, зокрема у системі MatLab, не реалізовані функції для здійснення алгебраїчних операцій у полях Галуа і програмістам, у разі необхідності, треба реалізовувати ці функції власноручно?

100. Поясніть коди комп'ютерних програм, призначених для виконання алгебраїчних операцій у полях Галуа різних порядків, які наведені у додатку Н. Які особливості написання цих програм Ви можете назвати?

101. Які засоби програмування системи науково-технічних розрахунків MatLab є найбільш зручними для реалізації функцій, призначених для виконання алгебраїчних операцій у полях Галуа? Свою відповідь обґрунтуйте.

102. Чому алгебраїчну операцію сумування елементів поля Галуа зазвичай зручно виконувати через подання елементів поля у вигляді двійкових матриць? Свою відповідь обґрунтуйте з точки зору теорії полів Галуа.

103. Поясніть приклад 3.31 та рис. 3.29.

104. Які поліноміальні операції у полях Галуа $GF(2^m)$ Вам відомі та як ці операції виконуються? Наведіть власні приклади виконання поліноміальних операцій у полях Галуа $GF(2^m)$.

105. Поясніть співвідношення (3.214) – (3.218).

106. Поясніть приклади 3.32 – 3.36.

107. Поясніть коди комп'ютерних програм, призначених для виконання поліноміальних операцій у полях Галуа різних порядків, які наведені у додатку Н. Які особливості реалізації цих програм Ви можете назвати?

108. Які засоби програмування системи науково-технічних розрахунків MatLab є найбільш зручними для виконання поліноміальних операцій у полях Галуа? Свою відповідь обґрунтуйте.

109. Яким чином у функціях, призначених для виконання поліноміальних операцій у полях Галуа, використовуються функції, призначені для виконання алгебраїчних операцій? Як таке використання пов'язане із принципом модульного програмування системи MatLab? Свою відповідь обґрунтуйте з точки зору теорії програмування.

110. Який Вам відомий спосіб формування твірних поліномів, призначених для створення систематичних кодів Ріда – Соломона? Наведіть власні приклади створення твірного поліному для коду Ріда – Соломона.

111. Від яких параметрів коду Ріда – Соломона залежить степінь твірного поліному? Свою відповідь обґрунтуйте.

112. Поясніть приклад 3.37.

113. Поясніть блок-схему алгоритму формування кодів Ріда – Соломона, яка наведена на рис. 3.36.

114. Поясніть приклади 3.38 – 3.40.

115. Поясніть код комп'ютерної програми, призначеної для формування кодів Ріда – Соломона, яка наведена у додатку О. Які особливості реалізації цієї програми Ви можете назвати?

116. У чому полягає сутність алгоритму Берлекемпа – Мессі, призначеного для декодування кодів Ріда – Соломона? Назвіть послідовні кроки цього алгоритму.

117. Поясніть співвідношення (3.220) – (3.223).

118. Чи існують у полях у полях Галуа $GF(2^m)$ елементи, які відповідають від'ємним числам у звичайній арифметиці? Як у полях Галуа $GF(2^m)$ визначається алгебраїчна операція піднесення до від'ємної степені? Поясніть співвідношення (3.224). Наведіть власні приклади обчислення функції від від'ємної степені для різних полів Галуа $GF(2^m)$.

119. Поясніть блок-схему алгоритму, яка наведена на рис. 3.41.

120. Поясніть теорему 3.11 та співвідношення (3.225).

121. Як для декодування послідовностей коду Ріда – Соломона використовується теорема Форні? Поясніть співвідношення (3.226) – (3.231) .

122. Які обмеження накладаються на параметри групових кодів Ріда – Соломона та чим ці обмеження обумовлені? Свою відповідь обґрунтуйте з точки зору загальних принципів теорії кодування сигналів.

123. Поясніть співвідношення (3.232), (3.233).

124. Поясніть властивості 3.1 та 3.2. Чому властивість 3.1 є більш строгою, а властивість 3.2 вважається її наслідком? Свою відповідь обґрунтуйте з точки зору загальних принципів теорії кодування сигналів.

125. Поясніть приклад 3.42.

126. Що являють собою синдроми помилок у кодах Ріда – Соломона та як вони обчислюються? Наведіть власні приклади обчислення синдромів помилок.

127. Що являє собою поліном локаторів помилок у кодах Ріда – Соломона та як обчислюються його коефіцієнти? Наведіть власні приклади обчислення коефіцієнтів поліному локаторів помилок.

128. Яким чином, за умови відомого поліному локаторів помилок, у кодах Ріда – Соломона знаходять розряди, в яких виникла помилка? Наведіть власні приклади знаходження помилкових розряді у послідовностях коду Ріда – Соломона.

129. Яким чином формується поліном величин помилок $\Omega(x)$? Наведіть власні приклади формування поліному величин помилок.

130. Яким чином обчислюються значення помилок у кодах Ріда – Соломона? Наведіть власні приклади обчислення значень помилок.

131. Яким чином формується вектор помилок в процесі декодування кодів Ріда – Соломона? Наведіть власні приклади формування вектору помилок.

132. Яким чином, за умови відомого вектору помилок, виправляються помилки у спотворених послідовностях кодів Ріда – Соломона? Поясніть співвідношення (3.223) та наведіть власні приклади виправлення помилок у кодах Ріда – Соломона.

133. Чому у випадку, коли заздалегідь відомо, що у коді Ріда – Соломона виникла лише одна помилка, алгоритм її пошуку значно спрощується? Свою відповідь обґрунтуйте з точки зору теорії завадостійкого кодування сигналів.

134. Поясніть співвідношення (3.235) – (3.241) та наведіть власний приклад пошуку однієї помилки у коді Ріда – Соломона.

135. Чому для декодування коду Ріда – Соломона, який був сформований для виявлення максимальної кількості помилок t, не може бути використаний спрощений алгоритм декодування, призначений для виявлення помилок меншої кратності τ? Свою відповідь обґрунтуйте з точки зору теорії завадостійкого кодування сигналів.

136. Яким чином у кодах Ріда – Соломона, сформованих для виявлення максимальної кількості помилок t, виявляються помилки меншої кратності $\tau < t$? Свою відповідь обґрунтуйте з точки зору теорії завадостійкого кодування сигналів.

137. Поясніть приклад 3.43.

138. Назвіть головні переваги групових завадостійких кодів над лінійними та циклічними кодами, завдяки яким групові коди знаходять

широке впровадження у сучасній кодувальній електронній апаратурі.

139. Для заданих послідовностей **C** у полі Галуа $GF(2^5)$ побудувати два типи кодів Ріда – Соломона: код, який виправляє одиничні помилки та код, який виправляє подвійні помилки. Занести до отриманих кодових послідовностей одиничну помилку до розряду n_1 та подвійну помилку до розрядів n_1 та n_2. Декодувати отримані спотворені кодові послідовності та виправити помилки у кодових комбінаціях. Визначити, у якому випадку виправити помилки не можливо та пояснити чому.

а) $\mathbf{C} = [14, 25, 17, 3, 11]$, $n_1 = 4$, $n_2 = 7$; б) $\mathbf{C} = [13, 5, 27, 12, 18]$, $n_1 = 3$, $n_2 = 5$;

в) $\mathbf{C} = [4, 2, 13, 15, 24]$, $n_1 = 2$, $n_2 = 4$; г) $\mathbf{C} = [7, 14, 14, 2, 25]$, $n_1 = 3$, $n_2 = 6$;

д) $\mathbf{C} = [15, 17, 14, 5, 28]$, $n_1 = 3$, $n_2 = 5$; е) $\mathbf{C} = [19, 17, 15, 15, 29]$, $n_1 = 2$, $n_2 = 7$;

є) $\mathbf{C} = [5, 27, 31, 15, 28]$, $n_1 = 3$, $n_2 = 4$; ж) $\mathbf{C} = [9, 27, 25, 25, 13]$, $n_1 = 4$, $n_2 = 7$;

з) $\mathbf{C} = [25, 27, 30, 15, 18]$, $n_1 = 4$, $n_2 = 5$; і) $\mathbf{C} = [1, 27, 5, 15, 23]$, $n_1 = 5$, $n_2 = 7$;

к) $\mathbf{C} = [14, 23, 31, 25, 8]$, $n_1 = 1$, $n_2 = 6$; л) $\mathbf{C} = [15, 26, 15, 5, 22]$, $n_1 = 4$, $n_2 = 5$.

Для виконання алгебраїчних та поліноміальних операцій у полі Галуа $GF(2^5)$ використовувати комп'ютерні програми, які наведені у додатку Н. Перевірити отримані результати з використанням програми **RSC**, код якої наведений у додатку О.

140. У чому полягає метод Гаусса – Зейделя, який застосовується для розв'язування систем лінійних алгебраїчних рівнянь у полях Галуа $GF(2^m)$? Наведіть власні приклади розв'язування систем лінійних алгебраїчних рівнянь у полях Галуа $GF(2^m)$ з використанням методу Гаусса – Зейделя.

141. Поясніть приклад 3.44.

142. З використанням метода Гаусса – Зейделя розв'язати наведені нижче системи рівнянь у полі Галуа $GF(2^5)$. Для виконання алгебраїчних операцій у полі Галуа $GF(2^5)$ використовувати комп'ютерні програми, які наведені у додатку Н. У яких випадках наведені системи рівнянь не можливо розв'язати? Поясніть чому.

а) $\begin{bmatrix} 23 & 12 \\ 18 & 3 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 10 \\ 15 \end{bmatrix}$; б) $\begin{bmatrix} 3 & 7 \\ 5 & 9 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 24 \\ 31 \end{bmatrix}$; в) $\begin{bmatrix} 13 & 27 \\ 25 & 19 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 17 \end{bmatrix}$;

г) $\begin{bmatrix} 16 & 28 \\ 18 & 31 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 17 \\ 19 \end{bmatrix}$; д) $\begin{bmatrix} 23 & 27 \\ 25 & 29 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 7 \\ 5 \end{bmatrix}$; е) $\begin{bmatrix} 3 & 7 \\ 1 & 30 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 14 \\ 17 \end{bmatrix}$;

є) $\begin{bmatrix} 26 & 3 \\ 18 & 17 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 27 \\ 23 \end{bmatrix}$; ж) $\begin{bmatrix} 16 & 13 \\ 28 & 7 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 7 \\ 2 \end{bmatrix}$; з) $\begin{bmatrix} 25 & 30 \\ 28 & 27 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 27 \\ 22 \end{bmatrix}$;

і) $\begin{bmatrix} 13 & 17 \\ 28 & 31 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 7 \\ 5 \end{bmatrix}$; к) $\begin{bmatrix} 29 & 14 \\ 28 & 17 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 19 \\ 24 \end{bmatrix}$; л) $\begin{bmatrix} 5 & 3 \\ 8 & 7 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 7 \\ 6 \end{bmatrix}$.

143. Що являють собою мінори та алгебраїчні доповнення для матриць, заданих у полях Галуа $GF(2^m)$? Поясніть співвідношення (3.243) – (3.245). Наведіть власні приклади обчислення мінорів та алгебраїчних доповнень для матриць, заданих у полях Галуа $GF(2^m)$.

144. Як у полях Галуа $GF(2^m)$ обчислюються матриці, зворотні до заданої? Наведіть власні приклади обчислення зворотних матриць у полях Галуа $GF(2^m)$.

145. У чому полягає метод обчислення визначників та обернених матриць, який застосовується для розв'язування систем лінійних алгебраїчних рівнянь у полях Галуа $GF(2^m)$? Наведіть власні приклади розв'язування систем лінійних алгебраїчних рівнянь у полях Галуа $GF(2^m)$ з використанням цього методу.

146. Знайдіть у полі Галуа $GF(2^5)$ матриці, обернені до визначених матриць другого порядку. Для виконання алгебраїчних операцій у полі Галуа $GF(2^5)$ використовувати комп'ютерні програми, які наведені у додатку Н. У яких випадках обернену матрицю не можливо обчислити? Поясніть чому. З використанням обчислених обернених матриць розв'язати системи лінійних рівнянь, які були наведені у завданні 142.

а) $\begin{bmatrix} 23 & 12 \\ 18 & 3 \end{bmatrix}$; б) $\begin{bmatrix} 3 & 7 \\ 5 & 9 \end{bmatrix}$; в) $\begin{bmatrix} 13 & 27 \\ 25 & 19 \end{bmatrix}$; г) $\begin{bmatrix} 16 & 28 \\ 18 & 31 \end{bmatrix}$; д) $\begin{bmatrix} 23 & 27 \\ 25 & 29 \end{bmatrix}$; е) $\begin{bmatrix} 3 & 7 \\ 1 & 30 \end{bmatrix}$;

є) $\begin{bmatrix} 26 & 3 \\ 18 & 17 \end{bmatrix}$; ж) $\begin{bmatrix} 16 & 13 \\ 28 & 7 \end{bmatrix}$; з) $\begin{bmatrix} 25 & 30 \\ 28 & 27 \end{bmatrix}$; і) $\begin{bmatrix} 13 & 17 \\ 28 & 31 \end{bmatrix}$; к) $\begin{bmatrix} 29 & 14 \\ 28 & 17 \end{bmatrix}$; л) $\begin{bmatrix} 5 & 3 \\ 8 & 7 \end{bmatrix}$.

147. Знайдіть у полі Галуа $GF(2^5)$ матриці, обернені до визначених матриць третього порядку. Для виконання алгебраїчних операцій у полі Галуа $GF(2^5)$ використовувати комп'ютерні програми, які наведені у додатку Н. У яких

випадках обернену матрицю не можливо обчислити? Поясніть чому.

а) $\begin{bmatrix} 25 & 13 & 17 \\ 24 & 31 & 8 \\ 16 & 12 & 15 \end{bmatrix}$; б) $\begin{bmatrix} 3 & 5 & 8 \\ 15 & 17 & 13 \\ 6 & 12 & 18 \end{bmatrix}$; в) $\begin{bmatrix} 13 & 15 & 8 \\ 15 & 17 & 13 \\ 26 & 12 & 28 \end{bmatrix}$; г) $\begin{bmatrix} 29 & 25 & 18 \\ 5 & 17 & 3 \\ 16 & 22 & 28 \end{bmatrix}$;

д) $\begin{bmatrix} 13 & 15 & 18 \\ 8 & 3 & 25 \\ 16 & 12 & 15 \end{bmatrix}$; е) $\begin{bmatrix} 13 & 15 & 28 \\ 15 & 27 & 23 \\ 26 & 12 & 18 \end{bmatrix}$; є) $\begin{bmatrix} 23 & 25 & 26 \\ 5 & 17 & 3 \\ 16 & 22 & 28 \end{bmatrix}$; ж) $\begin{bmatrix} 19 & 15 & 18 \\ 15 & 27 & 31 \\ 26 & 22 & 18 \end{bmatrix}$;

з) $\begin{bmatrix} 7 & 8 & 2 \\ 10 & 9 & 11 \\ 6 & 2 & 5 \end{bmatrix}$; і) $\begin{bmatrix} 3 & 5 & 18 \\ 25 & 17 & 13 \\ 16 & 12 & 18 \end{bmatrix}$; к) $\begin{bmatrix} 13 & 15 & 16 \\ 15 & 17 & 13 \\ 16 & 12 & 18 \end{bmatrix}$; л) $\begin{bmatrix} 29 & 25 & 28 \\ 25 & 27 & 21 \\ 26 & 22 & 28 \end{bmatrix}$.

148. З використанням комп'ютерних програм, призначених для виконання алгебраїчних операцій у полях Галуа $GF(2^m)$, які наведені у додатку Н, написати власну програму, яка обчислює визначники матриць другого та третього порядку у полях Галуа $GF(2^m)$. Із застосуванням написаної програми перевірити результати, які були отримані у завданнях 146 та 147.

149. У яких випадках для декодування послідовностей кодів Ріда – Соломона зручніше застосовувати матричні алгоритми, а у яких – алгоритм Берлекемпа – Мессі? Які критерії оцінки необхідно використовувати для порівняння цих алгоритмів? Свою відповідь обґрунтуйте з точки зору теорії завадостійкого кодування та теорії полів Галуа.

150. Поясніть структуру комп'ютерної програми **RSC**, призначеної для формування кодів Ріда – Соломона та декодування їхніх послідовностей. Які програмні засоби, призначені для виконання алгебраїчних операцій у полях Галуа $GF(2^m)$, та які стандартні засоби програмування системи MatLab були використані для написання цієї програми.

151. Як час виконання завдань формування та декодування кодів Ріда – Соломона з використанням комп'ютерних програмних засобів залежить від параметрів коду? Який спосіб проведення оцінок часу роботи програмних засобів, реалізований у системі MatLab, Вам відомий? Поясніть результати роботи програми **TESTRSC**, код якої наведений у додатку О.

152. Які особливості формування двійкових кодів Ріда – Соломона та декодування їхніх послідовностей Вам відомі? Поясніть числові дані, які наведені у таблицях 3.19 – 3.21, з точки зору теорії полів Галуа.

153. Поясніть приклади 3.45 та 3.46.

154. Для заданих послідовностей із трьох символів **C** побудувати у полі Галуа $GF(2^3)$ двійковий код Ріда – Соломона, який виправляє подвійну помилку. Занести до отриманих кодових послідовностей помилки до розрядів із заданими номерами n_1 та n_2. Декодувати отримані спотворені кодові послідовності та виправити помилки у кодових комбінаціях. Перевірити отримані результати з використанням програми **RSC**, код якої наведений у додатку О.

а) $\mathbf{C} = [3, 6, 2]$, $n_1 = 3$, $n_2 = 5$; б) $\mathbf{C} = [2, 2, 5]$, $n_1 = 2$, $n_2 = 6$;

в) $\mathbf{C} = [1, 0, 2]$, $n_1 = 6$, $n_2 = 7$; г) $\mathbf{C} = [2, 7, 1]$, $n_1 = 5$, $n_2 = 6$;

д) $\mathbf{C} = [1, 7, 0]$, $n_1 = 4$, $n_2 = 6$; е) $\mathbf{C} = [3, 5, 7]$, $n_1 = 2$, $n_2 = 6$;

є) $\mathbf{C} = [1, 1, 0]$, $n_1 = 5$, $n_2 = 7$; ж) $\mathbf{C} = [1, 3, 7]$, $n_1 = 4$, $n_2 = 6$;

з) $\mathbf{C} = [5, 1, 6]$, $n_1 = 4$, $n_2 = 7$; і) $\mathbf{C} = [3, 0, 6]$, $n_1 = 6$, $n_2 = 7$;

к) $\mathbf{C} = [0, 7, 5]$, $n_1 = 5$, $n_2 = 6$; л) $\mathbf{C} = [1, 7, 4]$, $n_1 = 5$, $n_2 = 7$.

155. Назвіть головні переваги та недоліки апаратного та програмного способів формування кодів Ріда – Соломона та декодування їхніх послідовностей. Свою відповідь обґрунтуйте з точки зору теорії завадостійкого кодування сигналів та теорії полів Галуа.

156. Чому спосіб програмного декодування кодів Ріда – Соломона не завжди можна реалізувати у сучасній швидкодійній цифровій електронній апаратурі? Свою відповідь обґрунтуйте з точки зору теорії завадостійкого кодування сигналів та теорії полів Галуа.

157. На яких загальних принципах будуються цифрові електронні схеми, призначені для формування кодів Ріда – Соломона?

158. Поясніть принципи роботи електронних кодувальних пристроїв, принципові схеми яких наведені на рис. 3.45 та 3.46.

159. Які програмні засоби використовуються у сучасній електронній апаратурі для програмування цифрових пристроїв, призначених для декодування послідовностей кодів Ріда – Соломона? Які головні вимоги

висуваються до таких програмних засобів та які вони мають функціональні можливості?

160. Поясніть принципи роботи електронних декодувальних пристроїв, принципові схеми яких наведені на рис. 3.47 – 3.50.

161. На основі яких теоретичних засад та базових положень зазвичай проводяться оцінки імовірності правильного та хибного прийманні послідовностей кодів Ріда – Соломона? Поясніть співвідношення (3.258) – (3.261).

162. Як можна оцінити максимальну кількість помилок, які можливо виправити з використанням кодів Ріда – Соломона, за умови відомої довжини кодової комбінації n? Поясніть співвідношення (3.262).

163. Поясніть графічні залежності, які наведені на рис. 3.51 та 3.52.

164. Як можна оцінити максимальну кількість помилок, які можливо виправити з використанням кодів Ріда – Соломона, за умови заданого порядку поля Галуа m. Поясніть співвідношення (3.263).

165. Поясніть код програми **RSERROR**, яка наведена у додатку О.

166. Які способи поліпшення коректувальної здатності кодів Ріда – Соломона Вам відомі і як вони можуть бути використані на практиці під час проектування реальних електронних систем? Наведіть власні приклади такого використання.

167. З використанням програми **RSERROR** побудувати графічні залежності $P_{\text{x}}(p_b)$ для визначених параметрів коду Ріда – Соломона та проаналізувати отримані результати моделювання. Оцінити максимальну кількість помилок, які може виправляти код, для найбільшого заданого значення кількості інформаційних символів k. Побудувати графічну залежність $P_{\text{x}}(p_b)$ для такого коду та порівняти її із іншими отриманими залежностями.

а) $m = 5$; $k = 6$, $t = 5$; $k = 8$, $t = 10$; $k = 10$, $t = 8$.

б) $m = 6$; $k = 10$, $t = 5$; $k = 15$, $t = 10$; $k = 15$, $t = 20$.

в) $m = 7$; $k = 15$, $t = 6$; $k = 20$, $t = 12$; $k = 25$, $t = 30$.

г) $m = 4$; $k = 5$, $t = 3$; $k = 6$, $t = 4$; $k = 4$, $t = 5$.

д) $m = 8$; $k = 20$, $t = 15$; $k = 30$, $t = 20$; $k = 40$, $t = 30$.

е) $m = 7$; $k = 25$, $t = 10$; $k = 30$, $t = 15$; $k = 40$, $t = 20$.

є) $m = 6$; $k = 5, t = 15$; $k = 20, t = 10$; $k = 10, t = 25$.

ж) $m = 5$; $k = 3, t = 10$; $k = 4, t = 12$; $k = 5, t = 13$.

з) $m = 5$; $k = 4, t = 8$; $k = 6, t = 10$; $k = 7, t = 12$.

і) $m = 7$; $k = 20, t = 7$; $k = 25, t = 10$; $k = 30, t = 25$.

к) $m = 8$; $k = 25, t = 10$; $k = 30, t = 25$; $k = 50, t = 50$.

л) $m = 6$; $k = 10, t = 5$; $k = 15, t = 10$; $k = 20, t = 18$.

168. З використанням співвідношень (3.24) розрахувати параметри надлишковості для кодів Ріда – Соломона із кількістю інформаційних символів та коректувальними параметрами, заданими у завданнях 139, 154 та 167.

169. Що являють собою каскадні коди та як вони використовуються у сучасній цифровій електронній апаратурі? Наведіть власні приклади такого використання каскадних кодів.

170. Поясніть схему каналу зв'язку з каскадним кодуванням інформації, наведену на рис. 3.53.

171. Поясніть визначення 3.13.

172. Які параметри мають каскадні коди та як вони обчислюються? Наведіть власні приклади розрахунку параметрів каскадного коду.

173. Поясніть співвідношення (3.266) та (3.267).

174. Який алгоритм побудови каскадних кодів Вам відомий? Наведіть власні приклади використання цього алгоритму.

175. Поясніть співвідношення (3.268), (3.269) та (3.270).

176. Поясніть узагальнену структуру каскадного коду, яка відображена на рис. 3.54.

177. Як обчислюється коректувальна здатність каскадного коду? Наведіть власні приклади визначення цього параметра каскадного коду.

178. Поясніть співвідношення (3.271).

179. Поясніть приклад 3.47.

180. Поясніть співвідношення (3.272), (3.273) та (3.274).

181. Поясніть приклад 3.48.

182. Побудуйте каскадні коди для заданих кодових послідовностей, заданих в таблиці 3.34, та розрахуйте їхню коректувальну здатність. Спосіб формування зовнішнього та внутрішнього коду вказаний для кожного з варіантів окремо.

Таблиця 3.34 – Інформаційні послідовності та способи формування зовнішніх та внутрішніх кодів для завдання 182

№ варіанту	Інформаційне слово	Внутрішній код	Зовнішній код
1.	1 2 3 6 5 3 7 8 5 3 4 2 1 0 4 0 1 2 6 4 8 3 4 2 4 2 8 2 6 2 8 3 0 5 0 3 4 2 3 6 1 6 2 4 0 4 0 2 3 1 7 3 0 7 9 1 2 0 4 4 5 2 3 4 1 3 2 4 5 0 9 3	РС-код (10, 4) над полем Галуа $GF(2^4)$	РС-код (26, 18) над полем Галуа $GF(2^5)$
2.	1 2 3 6 7 3 3 8 5 3 0 2 1 9 4 0 3 4 8 3 4 7 5 2 8 2 6 2 3 9 0 5 8 2 3 6 1 6 2 4 4 4 0 2 8 1 3 3 1 7 2 3 2 7 1 6 4 0 3 0 5 3 8 4	РС-код (12, 4) над полем Галуа $GF(2^4)$	РС-код (20, 16) над полем Галуа $GF(2^5)$
3.	9 3 6 6 5 2 7 1 5 8 4 6 1 9 4 0 0 3 1 8 2 8 3 4 5 8 3 4 6 3 2 4 5 5 5 7 4 2 0 6 0 6 4 4 0 4 0 2 3 1 4 3 0 9 6 0 3 4 5 9 0 0 4 5 9 7 5 0 0 0 7 2	РС-код (8, 4) над полем Галуа $GF(2^4)$	РС-код (28, 18) над полем Галуа $GF(2^5)$
4.	4 2 8 2 3 3 3 2 9 7 3 2 0 1 9 0 8 0 8 3 2 7 0 3 4 9 3 4 5 3 0 1 2 3 1 4 1 6 0 4 5 2 9 9 3 5 9 5 1 7 2 3 2 7 1 6 4 0 3 0 5 3 8 4	РС-код (6, 4) над полем Галуа $GF(2^4)$	РС-код (22, 16) над полем Галуа $GF(2^5)$
5.	3 9 2 4 2 0 4 5 5 3 6 3 9 9 3 0 2 2 9 1 9 5 7 3 9 5 4 2 3 2 8 4 5 0 3 7 1 9 3 7 0 2 3 4 9 9 0 6 3 9 2 1 7 4 5 2 3 8 2 0 2 8 3 0 0 6 1 1 5 0 0 0	РС-код (10, 4) над полем Галуа $GF(2^4)$	РС-код (20, 18) над полем Галуа $GF(2^5)$
6.	0 9 8 9 5 7 2 2 6 0 1 8 0 3 9 6 8 0 8 3 2 7 0 3 4 9 3 4 5 3 0 1 1 9 1 4 8 0 0 7 0 0 4 6 3 6 2 1 1 2 7 8 3 2 9 2 1 5 0 3 6 3 8 1	РС-код (12, 4) над полем Галуа $GF(2^4)$	РС-код (24, 16) над полем Галуа $GF(2^5)$

Таблиця 3.34 – Інформаційні послідовності та способи формування зовнішніх та внутрішніх кодів для завдання 182 (закінчення)

№ варіанту	Інформаційне слово	Внутрішній код	Зовнішній код
7.	3 0 2 2 5 3 9 2 4 2 0 4 5 3 6 3 9 9 5 2 9 1 9 5 7 8 4 3 9 5 4 2 5 0 3 7 9 0 6 3 9 1 2 1 7 4 9 3 7 0 2 3 4 9 5 2 3 8 3 0 0 6 1 1 5 2 0 0 0 0 2 8	РС-код (8, 4) над полем Галуа $GF(2^4)$	РС-код (28, 18) над полем Галуа $GF(2^5)$
8.	5 7 2 2 6 1 8 0 0 0 3 9 6 9 8 9 0 3 4 9 3 8 0 8 3 2 7 4 5 3 0 1 4 1 9 1 4 8 0 0 7 0 0 6 3 6 2 1 2 9 2 1 5 1 2 7 8 3 0 3 6 3 8 1	РС-код (8, 4) над полем Галуа $GF(2^4)$	РС-код (20, 16) над полем Галуа $GF(2^5)$
9.	6 3 9 9 3 2 4 2 0 0 9 4 5 3 2 2 5 3 7 8 0 3 0 5 5 2 9 1 9 5 4 2 5 0 3 7 2 1 7 4 9 3 9 0 6 3 9 1 7 0 2 3 4 9 5 2 3 8 5 2 3 6 1 1 0 0 0 0 0 0 2 8	РС-код (10, 4) над полем Галуа $GF(2^4)$	РС-код (24, 18) над полем Галуа $GF(2^5)$
10.	6 9 8 9 5 8 0 0 0 3 9 7 2 2 6 1 3 2 7 4 5 6 3 4 0 0 8 0 8 3 0 1 8 4 1 0 1 0 0 0 6 3 6 2 1 7 0 0 1 2 7 8 3 0 2 9 2 1 5 3 6 3 8 1	РС-код (12, 4) над полем Галуа $GF(2^4)$	РС-код (22, 16) над полем Галуа $GF(2^5)$
11.	2 0 0 9 4 5 6 3 9 9 3 2 4 3 2 2 5 3 5 2 9 1 9 5 4 2 5 7 8 0 3 0 5 0 3 7 2 1 7 3 9 1 7 0 4 9 3 9 0 6 2 3 4 9 5 1 8 0 0 0 0 0 0 2 8 2 3 8 5 9 3 6	РС-код (14, 4) над полем Галуа $GF(2^4)$	РС-код (28, 18) над полем Галуа $GF(2^5)$
12.	6 9 8 7 2 2 6 1 9 5 8 0 0 0 3 9 3 4 0 0 8 3 2 7 4 5 6 0 8 3 0 1 1 0 1 6 2 1 7 0 0 0 0 0 6 3 8 4 6 3 8 1 3 0 2 9 2 1 1 2 7 8 5 3	РС-код (10, 4) над полем Галуа $GF(2^4)$	РС-код (24, 16) над полем Галуа $GF(2^5)$

183. Виконати завдання 182 з використанням програми, наведеної у додатку О. Перевірити з використанням цієї програми коректність роботи створеного каскадного коду та визначити його коректувальну здатність. Для виконання цього завдання написати власну програму мовою програмування системи науково-технічних розрахунків MatLab.

184. Як визначається дискретне перетворення Фур'є у полі Галуа? Наведіть власні приклади визначення такого перетворення та розрахунку відповідних коефіцієнтів.

185. Поясніть співвідношення (3.275) та (3.276).

186. Що являє собою характеристика поля Галуа порядку q та як вона визначається? Наведіть власні приклади визначення характеристики поля Галуа.

187. Поясніть визначення 3.14.

188. Поясніть властивість 3.3 та співвідношення (3.277), (3.278).

189. Поясніть властивості 3.4 та 3.5. Наведіть приклади використання цих властивостей.

190. Поясніть співвідношення (3.279) – (3.282).

191. Що являє собою спектральний вектор для дискретного перетворення Фур'є у полі Галуа порядку q? Наведіть власні приклади формування такого вектору.

192. Поясніть визначення 3.15.

193. Поясніть властивості 3.7 та 3.8.

194. Що являють собою спряжені елементи у полі Галуа порядку q? Поясніть визначення 3.16 та властивість 3.9.

195. Поясніть співвідношення (3.283) – (3.285).

196. Що являють собою сліди у теорії кодування та як вони використовуються для кодування дискретних сигналів. Наведіть власні приклади формування слідів.

197. Поясніть визначення 3.17 та 3.18.

198. Поясніть співвідношення (3.286)

199. Поясніть властивості 3.10 – 3.14 та співвідношення (3.287), (3.288).

200. Поясніть визначення 3.19.

201. Поясніть властивість 3.15 та співвідношення (3.289) – (3.292).

202. Поясніть приклад 3.49.

203. Що являють собою розширені коди Ріда – Соломона та які способи їхнього формування Вам відомі? Наведіть власні приклади формування таких кодів.

204. Поясніть співвідношення (3.294) та (3.295).

205. Поясніть визначення 3.20, 3.21 та 3.22.

206. Поясніть узагальнену схему кодеру, яка наведена на рис. 3.56.

207. Поясніть спосіб формування розширених кодів Ріда – Соломона із збільшеною кодовою відстанню та наведіть власні приклади формування таких кодів.

208. Поясніть співвідношення (3.296).

209. Що являють собою ітеративні коди та як вони використовуються у цифрових електронних пристроях? Наведіть власні приклади такого використання ітеративних кодів.

210. Поясніть співвідношення (3.296) та наведіть приклади його використання.

211. Поясніть узагальнену структуру ітеративного коду, наведену на рис. 3.57.

212. Поясніть співвідношення (3.297) та наведіть приклади їх використання.

213. Які помилки не можна виправляти з використанням ітеративного коду і чому? Свою відповідь обґрунтуйте.

214. Поясніть рис. 3.58.

215. Як оцінюється відносна кількість помилок, які можна виправляти з використанням ітеративного коду.

216. Поясніть співвідношення (3.298) – (3.300) та наведіть власні приклади оцінки коректувальної здатності ітеративного коду з використанням цих співвідношень.

217. Як оцінюється мінімальна вага ненульового вектору ітеративного коду? Наведіть власні приклади проведення таких оцінок.

218. Поясніть співвідношення (3.300) та наведіть власні приклади оцінки мінімальної ваги ненульового вектору ітеративного коду з використанням цього співвідношення.

219. Поясніть приклад 3.50.

220. Побудуйте ітеративний код розмірністю (144, 64) на основі коду Хеммінга (12,8) для заданих нижче восьми кодових послідовностей із восьми бітів:

а) 10101101, 10111101, 11101101, 11011101, 10100101, 10000101, 10000001, 11110000;

б) 10000001, 10101010, 11111000, 10111011, 11010001, 11011101, 11111111, 10000011;

в) 01011010, 11111110, 11111001, 11010101, 11010111, 01000010, 10101010, 00001111;

г) 10011001, 10111010, 11011010, 10101110, 10010101, 11010101, 11100111, 10011011;

д) 10111011, 10101110, 11010011, 10111110, 10110111, 11000101, 11000111, 11011011;

е) 10000000, 00101010, 11001000, 10101110, 11010101, 11010101, 11100111, 00000011;

є) 11011010, 11000110, 11001000, 01010101, 11010101, 11011010, 11101111, 10001111;

ж) 11011110, 11010111, 11001011, 11011101, 11011101, 11011110, 11101101, 10101111;

з) 10101011, 11101110, 11011010, 10101010, 11011101, 11010111, 10100111, 10010011;

і) 11111011, 11001110, 11001010, 10001110, 11010101, 11010101, 10100101, 10011011;

к) 10011011, 11001010, 11001110, 10101110, 11010111, 01010101, 00100101, 11011011.

Перевірити роботу сформованого ітеративного коду у разі наявності в ньому одиночної, подвійної, потрійної та чотирикратної помилки.

221. Написати з використанням засобів програмування системи MatLab власну комп'ютерну програму, призначену для формування ітеративного коду на основі коду Хеммінга (12, 8) та для дешифрування відповідних кодових послідовностей. Під час написання цієї програми формувати код Хеммінга з використанням програми **`Hamming_Coding`**, наведеної у додатку З, викликаючи її як зовнішню функцію із відповідними параметрами. У програмі передбачити можливості як створення відповідного ітеративного коду, так і його дешифрування із пошуком помилок. Для тестування написаної програми виконати з її допомогою завдання 220.

222. Як формується ітеративний код, який використовується для надійного запису цифрової інформації на магнітну стрічку? Наведіть структуру цього ітеративного коду та поясніть спосіб його побудови. Наведіть власні приклади формування такого ітеративного коду.

223. Поясніть рис. 3.59 та таблицю 3.22.

224. Поясніть співвідношення (3.302) – (3.307).

225. Поясніть приклад 3.51.

226. Побудувати ітеративний код, призначений для записування інформації на магнітну стрічку, вважаючи, що стрічка має 8 інформаційних

доріжок, кадр формується із 8 бітів, та в ньому записані бітові послідовності, задані у завданні 220. Перевірити роботу сформованого ітеративного коду у разі наявності одиночної, подвійної, потрійної та чотирикратної помилки на одній та на різних доріжках.

227. Написати з використанням засобів програмування системи MatLab власну комп'ютерну програму для формування ітеративного коду, призначеного для записування інформації на магнітну стрічку та для дешифрування відповідних кодових послідовностей. Під час написання цієї програми формувати систематичний циклічний код з використанням програми **CRC**, наведеної у додатку К, викликаючи її як зовнішню функцію із відповідними параметрами. У програмі передбачити можливості як створення відповідного ітеративного коду, так і його дешифрування із пошуком помилок. Для тестування написаної програми виконати з її допомогою завдання 226.

228. Чим суттєво відрізняється алгоритм роботи згорткових кодів від алгоритмів роботи лінійних та групових кодів? Свою відповідь обґрунтуйте та наведіть доречні приклади алгоритмів роботи згорткових кодів.

229. Наведіть відповідні приклади з художньої літератури та життєві ситуації, які безпосередньо пов'язані з принципом формування згорткового коду. Поясніть, у чому полягає взаємозв'язок між наведеними вами прикладами та принципом завадостійкого кодування.

230. Які параметри завадостійких кодів Вам відомі? Наведіть власні приклади обчислення та використання цих параметрів.

231. Що являє собою довжина кодового обмеження K для комбінації згорткового коду? Наведіть власні приклади визначення цього параметру.

232. Що являє собою параметр надлишковості n для комбінації згорткового коду? Наведіть власні приклади визначення цього параметру.

233. Поясніть принцип роботи узагальненої структурної схеми кодеру згорткового коду, яка наведена на рис. 3.60.

234. Поясніть принцип роботи схеми кодеру згорткового коду з параметрами $K = 3$ та $\frac{1}{n} = \frac{1}{2}$, яка наведена на рис. 3.61.

235. Поясніть приклад 3.52 та рис. 3.62.

236. Поясніть приклад 3.53 та рис. 3.63.

237. Проаналізуйте роботу схеми кодеру згорткового коду, наведеної на рис. 3.61, та знайдіть двійкові послідовності вихідних сигналів за умови надходження на вхід цієї схеми послідовностей 1 0 0, 0 1 1 та 1 1 1.

238. Поясніть приклад 3.54. Наведіть власні приклади обчислення параметрів згорткових кодів за відомою схемою кодеру.

239. Які способи подання згорткових кодів Вам відомі та як ці способи подання пов'язані між собою?

240. Що являє собою поліноміальне подання згорткових кодів та як воно використовується для аналізу особливостей роботи кодерів та декодерів згорткового коду? Наведіть власні приклади описання згорткових кодів через поліноміальне подання.

241. Поясніть приклад 3.55.

242. Що являє собою подання згорткових кодів у вигляді діаграми станів скінченного автомату та як воно використовується для аналізу особливостей роботи кодерів та декодерів згорткового коду? Наведіть власні приклади описання згорткових кодів через діаграму станів скінченного автомату.

243. Поясніть рис. 3.64.

244. Поясніть приклади 3.56 та 3.57.

245. Поясніть числові дані, які наведені у таблицях 3.25 та 3.26.

246. Що являє собою подання згорткових кодів у вигляді деревоподібної діаграми та як воно використовується для аналізу особливостей роботи кодерів та декодерів згорткового коду? Наведіть власні приклади описання згорткових кодів через деревоподібну діаграму.

247. Поясніть рис. 3.65.

248. Що являє собою подання згорткових кодів у вигляді ґраткової діаграми та як воно використовується для аналізу особливостей роботи кодерів та декодерів згорткового коду? Наведіть власні приклади описання згорткових кодів через ґраткову діаграму.

249. Поясніть рис. 3.66.

250. Поясніть рис. 3.67 та 3.68.

251. Поясніть приклад 3.58, рис. 3.69 та числові дані, наведені у таблиці 3.27.

252. Поясніть діаграму станів скінченного автомату, яка наведена на рис. 3.70.

253. Поясніть структуру ґраткової діаграми, яка наведена на рис. 3.71.

254. Поясніть приклади 3.59 та 3.60.

255. Що являє собою алгоритм послідовного декодування згорткових кодів та як він використовується на практиці для декодування сигналів у системах зв'язку та інформаційних електронних системах? Наведіть власні приклади використання цього алгоритму в інформаційних електронних системах.

256. Поясніть узагальнену блок-схему алгоритму послідовного декодування згорткового коду, яка наведена на рис. 3.72.

257. Поясніть рис. 3.73.

258. Що являє собою алгоритм декодування згорткових кодів із зворотним зв’язком та як він використовується на практиці для декодування сигналів у системах зв'язку та інформаційних електронних системах? Наведіть власні приклади використання цього алгоритму в інформаційних електронних системах.

258. Поясніть співвідношення (3.308) та числові дані, які наведені в таблиці 3.28.

259. Поясніть рис. 3.74.

260. Що являє собою алгоритм декодування згорткових кодів за принципом максимальної правдоподібності та як він використовується на практиці для декодування сигналів у системах зв'язку та інформаційних електронних системах? Наведіть власні приклади використання цього алгоритму в інформаційних електронних системах.

261. Поясніть співвідношення (3.309) – (3.312).

262. Поясніть ґраткову діаграму, наведену на рис. 3.75.

263. Поясніть рис. 3.76 та рис. 3.77.

264. Поясніть рис. 3.78.

265. У чому полягають головні переваги та недоліки алгоритму послідовного декодування згорткових кодів? Наведіть власні приклади, в яких проілюструйте вказані переваги та недоліки.

266. У чому полягають головні переваги та недоліки алгоритму декодування згорткових кодів Вітербі? Наведіть власні приклади, в яких проілюструйте вказані переваги та недоліки.

267. Поясніть графічні залежності, які наведені на рис. 3.79.

268. Що являє собою матричне подання згорткових кодів? Наведіть власні приклади використання матриць для формування згорткових кодів.

269. Поясніть визначення 3.23 та 3.24.

270. Поясніть співвідношення (3.313) – (3.320).

271. Поясніть приклад 3.61.

272. Поясніть співвідношення (3.321), (3.322).

273. Що являє собою код Вейнера – Еша? Наведіть власні приклади використання цього коду в інформаційних електронних системах.

274. Поясніть рис. 3.80 та 3.81.

275. Поясніть рис. 3.82 та 3.83.

286. Поясніть числові дані, які наведені в таблиці 3.29.

287. Поясніть співвідношення (3.323) – (3.328).

288. Поясніть властивості 3.16 та 3.17.

289. Поясніть приклад 3.63.

290. Поясніть співвідношення (3.329), (3.330).

291. Поясніть рис. 3.84 та рис. 3.85.

292. Що являє собою алгоритм Фано декодування згорткових кодів та як він пов'язаний з алгоритмом послідовного декодування? Наведіть власні приклади використання цього алгоритму в інформаційних електронних системах.

293. Поясніть співвідношення (3.331) – (3.337).

294. Поясніть узагальнену блок-схему алгоритму Фано, яка наведена на рис. 3.86.

295. Поясніть принцип роботи цифрового електронного пристрою, призначеного для декодування послідовностей згорткового коду з використанням алгоритму Фано, узагальнена структурна схема якого наведена на рис. 3.87.

296. Поясніть принцип роботи електричної схеми кодеру згорткового коду з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$, яка наведена на рис. 3.88.

297. Поясніть приклад 3.65.

298. Поясніть принцип роботи принципової електричної схеми кодеру згорткового коду з параметрами $K = 3$, $\frac{1}{n} = \frac{1}{2}$, яка наведена на рис. 3.89.

299. Поясніть рис. 3.90.

300. Поясніть співвідношення (3.338) – (3.340).

301. Поясніть рис. 3.91 – 3.93.

302. Що являє собою матриця станів згорткового коду та як вона формується? Наведіть власні приклади формування такої матриці.

303. Чому описання властивостей згорткових кодів в програмних засобах, призначених для декодування їхніх послідовностей, зручно проводити саме через аналіз матриці станів? Які лінгвістичні та алгоритмічні

засоби сучасних систем програмування використовуються для такого описання?

304. Наведіть власні приклади описання властивостей згорткових кодів через аналіз структури матриці станів з використанням програмних засобів системи науково-технічних розрахунків MatLab.

305. Поясніть властивості 3.18 – 3.25. Наведіть власні приклади використання цих властивостей для формування матриці станів згорткового коду.

306. Поясніть співвідношення (3.341) – (3.344). Як ці співвідношення пов’язані з математичною логікою та теорією предикатів? Наведіть приклади використання цих співвідношень для формування матриці станів згорткового коду.

307. Поясніть узагальнену структуру матриці станів згорткового коду, яка показана на рис. 3.94.

308. Поясніть приклад 3.66 та числові дані, які наведені у таблиці 3.30.

309. Поясніть приклад 3.67, співвідношення (3.345) та рис. 3.95.

310. Поясніть приклад 3.68.

311. Поясніть рис. 3.96.

312. Поясніть числові дані, які наведені у таблиці 3.31, та схему скінченного автомату, наведену на рис. 3.97.

313. Поясніть числові дані, які наведені у таблиці 3.32.

314. У чому полягає узагальнений алгоритм формування послідовностей згорткового коду? Наведіть власні приклади використання такого алгоритму.

315. Поясніть співвідношення (3.345) – (3.349). Наведіть власні приклади використання цих співвідношень.

316. Поясніть блок-схему узагальненого алгоритму формування послідовності згорткового коду, яка наведена на рис. 3.98.

317. У чому полягає складність завдання формування алгоритмів

пошуку помилок у згорткових кодах? На яких теоретичних засадах зазвичай базуються ці алгоритми? Наведіть власні приклади таких алгоритмів та поясніть їх сутність.

318. У чому полягає сутність алгоритм пошуку оптимального шляху за ґратковою діаграмою згорткового коду, який базується на аналізі структури матриці станів? Наведіть власні приклади використання цього алгоритму для розшифрування послідовностей згорткових кодів.

319. Поясніть співвідношення (3.350) – (3.352). Наведіть приклади використання цих співвідношень для пошуку найбільш правдоподібного шляху у кодовій комбінації згорткового коду.

320. Поясніть блок-схему алгоритму, яка наведена на рис. 3.100.

321. Що являють собою пакетні помилки, як вони виникають, та чому саме пошук пакетних помилок зазвичай є найбільш складним завданням під час розшифрування кодових послідовностей згорткових кодів? Наведіть власні приклади пошуку пакетних помилок.

322. У чому полягає сутність алгоритму виправлення помилок у згорткових кодах через пошук оптимального шляху із найменшою метрикою? Наведіть власні приклади використання цього алгоритму для розшифрування послідовностей згорткових кодів.

323. Що являє собою алгоритм пошуку мінімального шляху поточного стану (АПМШПС) та як він використовується для аналізу кодових комбінацій згорткових кодів. Наведіть власні приклади використання цього алгоритму для розшифрування послідовностей згорткових кодів.

324. Поясніть співвідношення (3.353) – (3.355). Наведіть приклади використання цих співвідношень для пошуку пакетних помилок у кодовій комбінації згорткового коду.

325. Поясніть блок-схему алгоритму, яка наведена на рис. 3.101.

326. Поясніть код програми, яка наведена у додатку П.

327. У чому полягають переваги використання програмних засобів та функцій системи науково-технічних розрахунків MatLab для написання комп’ютерної програми, призначеної для формування послідовностей згорткових кодів та їхнього декодування? Які інші відомі Вам програмні засоби можуть бути використані для вирішення такого завдання? Наведіть приклади завершених фрагментів програм, призначених для формування кодових послідовностей згорткових кодів та для пошуку оптимального шляху у сформованій кодовій комбінації.

328. Які змінні, функції та модулі використані в програмі, наведеній у додатку П? Поясніть загальну структуру цієї програми.

329. Поясніть блок-схему алгоритму, яка наведена на рис. 3.102.

330. Чому аналіз коректувальної здатності для згорткових кодів є значно більш складним завданням, ніж те саме завдання для лінійних та групових кодів? Свою відповідь обґрунтуйте та наведіть власні приклади аналізу коректувальної здатності згорткових кодів.

331. Чому під час аналізу коректувальної здатності згорткових кодів аналіз мінімальної відстані між кодовими комбінаціями завжди проводиться відносно нульової кодової комбінації. Свою відповідь обґрунтуйте та сформулюйте відпоівдні теореми теорії завадостійкого кодування.

332. Поясніть рис. 3.103.

333. Що являє собою мінімальний просвіт згорткового коду? Наведіть власні приклади обчислення мінімального просвіту для згорткового коду, який формується за заздалегідь заданим алгоритмом.

334. Поясніть визначення 3.25.

335. Поясніть співвідношення (3.356) та наведіть власні приклади обчислення коректувальної здатності згорткового коду.

336. Як для обчислення мінімального просвіту згортковго коду використовується подання алгоритму кодування через діаграму станів скінченного автомату? Наведіть власні приклади обчислення мінімального

просвіту згорткового коду з використанням теорії скінченних автоматів.

337. Поясніть діаграми станів скінченних автоматів, які наведені на рис. 3.104 та 3.105.

338. Поясніть співвідношення (3.357) – (3.366).

339. Поясніть діаграму станів скінченного автомату, яка наведена на рис. 3.106.

340. Поясніть співвідношення (3.367) – (3.373).

341. Які головні правила щодо формування передавальної функції декодеру згорткового коду Вам відомі? Наведіть власні приклади використання цих правил.

342. Поясніть приклад 3.69.

343. Поясніть схему скінченного автомату, яка наведена на рис. 3.107.

344. Поясніть співвідношення (3.374) – (3.402).

345. Поясніть приклад 3.70.

346. Поясніть схему скінченного автомату, яка наведена на рис. 3.108.

347. Поясніть співвідношення (3.402) – (3.458).

348. У чому полягають головні переваги ручних методів розрахунку мінімального просвіту згорткового коду? Наведіть власні приклади проведення таких розрахунків.

349. У чому полягають головні переваги комп’ютерних методів розрахунку мінімального просвіту згорткового коду? Наведіть власні приклади проведення таких розрахунків.

350. Які засоби системи науково-технічних розрахунків MatLab можуть бути ефективно використані для проведення операцій з поліномами? Наведіть власні приклади такого використання цих програмних засобів.

351. Як з використанням засобів програмування системи науково-технічних розрахунків MatLab можна суттєво зменшити кількість поліноміальних операції у разі необхідності опрацювання поліномів високих

порядків? Наведіть власні приклади такого використання цих засобів.

352. Поясніть співвідношення (3.459).

353. Як передавальна функція декодеру згорткового коду $T(D, L, N)$ може бути використана для обчислення імовірності бітової помилки у кодовій комбінації P_B? Наведіть власні приклади такого використання функції $T(D, L, N)$.

354. Як умови подання та поширення цифрового сигналу, який кодується згортковим кодом, впливають на імовірності бітової помилки у кодовій комбінації P_B? Свою відповідь обґрунтуйте та наведіть власні приклади впливу способу кодування сигналу на імовірність бітової помилки у кодовій комбінації P_B.

355. Як визначається імовірність бітової помилки у кодовій комбінації P_B за умови використання двійкових цифрових кодів? Наведіть власні приклади обчислення імовірність бітової помилки у кодовій комбінації P_B за такої умови.

356. Поясніть співвідношення (3.460).

357. Як залежить імовірність бітової помилки у кодовій комбінації P_B від параметрів згорткового коду? Свою відповідь обґрунтуйте та наведіть власні приклади обчислення імовірності бітової помилки у кодовій комбінації P_B для різних параметрів згорткового коду.

358. Поясніть співвідношення (3.461) – (3.468).

359. Поясніть графічні залежності, які наведені на рис. 3.109.

360. Як визначається імовірність бітової помилки у кодовій комбінації P_B для каналу з гауссовим шумом за умови фазової кодоімпульсної модуляції двійкового сигналу? Наведіть власні приклади обчислення імовірності бітової помилки у кодовій комбінації P_B за такої умови.

361. Поясніть співвідношення (3.469), (3.470).

362. Поясніть співвідношення (3.471) – (3.473).

363. Поясніть графічні залежності, які наведені на рис. 3.110.

364. Для структурних схем кодерів, які наведені на рис. 3.115, побудувати діаграму станів скінченного автомату, деревоподібну та ґраткову діаграму.

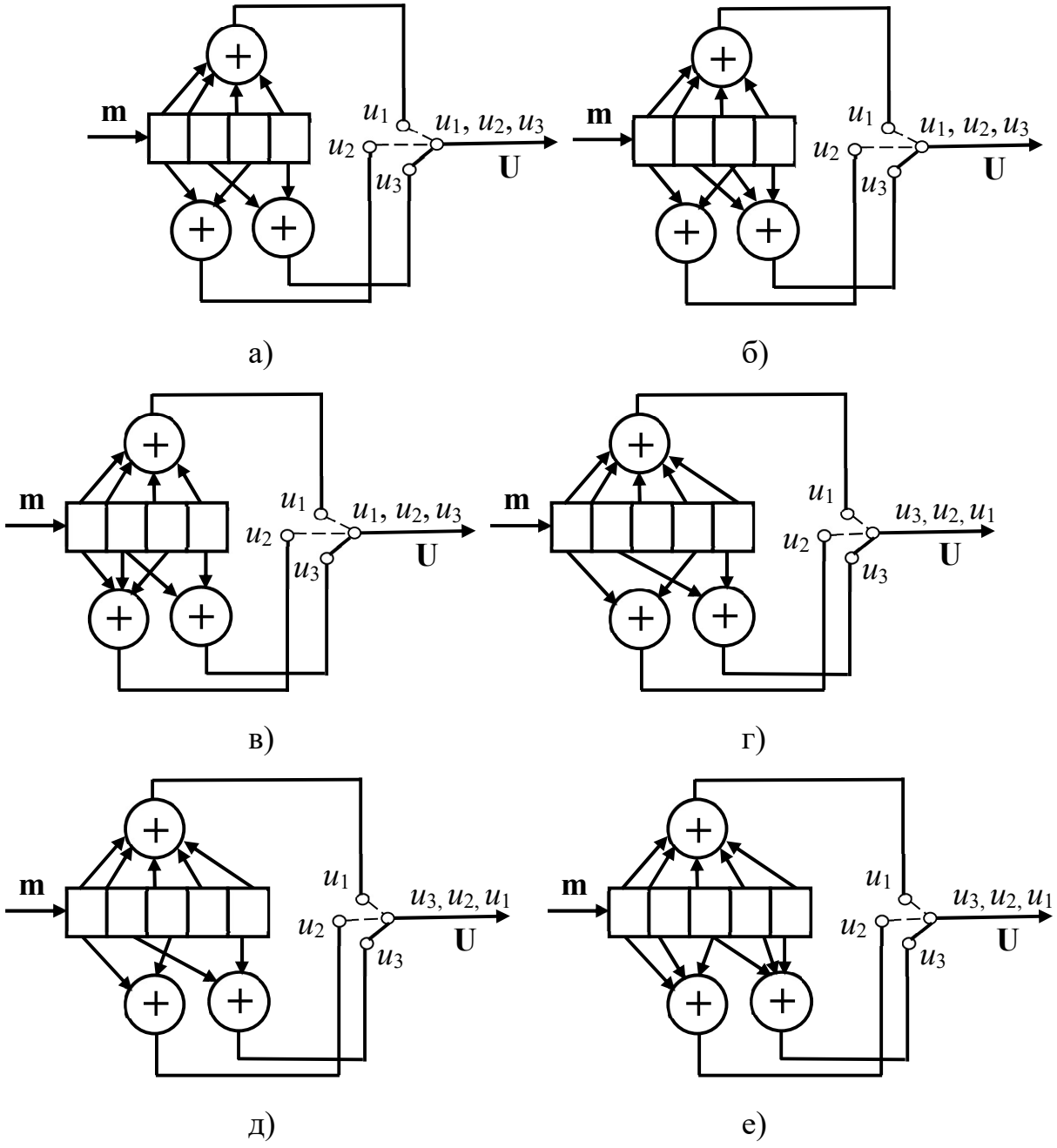


Рис. 3.115 Структурні схеми кодерів для завдань 364 – 370

365. Записати матрицю станів для згорткових кодів, які формуються з використанням структурних схем кодерів, наведених на рис. 3.115.

366. Для згорткових кодів, які формуються з використанням

структурних схем кодерів, наведених на рис. 3.115, знайти параметр мінімального просвіту та передавальну функцію.

367. Для згорткових кодів, які формуються з використанням структурних схем кодерів, наведених на рис. 3.115, побудувати залежності максимального значення імовірності помилки у кодовій комбінації від імовірності бітової помилки, за умови застосування жорсткої схеми кодування.

368. Для згорткових кодів, які формуються з використанням структурних схем кодерів, наведених на рис. 3.115, побудувати залежності максимального значення імовірності помилки у кодовій комбінації від імовірності бітової помилки, за умови поширення двійкового сигналу з фазовою кодоімпульсною модуляцією у каналі зв'язку з гауссовим шумом.

369. З використанням структурних схем кодерів, наведених на рис. 3.115, закодувати кодову послідовність **m**. Занести пакетну помилку в розряди кодової комбінації сформованого згорткового коду, від початкового, n_1, до кінцевого, n_2, та показати можливість виправлення такої помилки. Тип кодеру, а також значення **m**, n_1 та n_2, взяти для відповідного варіанту завдання з таблиці 3.35. У яких випадках завдання поставлено некоректно та чому? Обґрунтуйте свою відповідь.

Таблиця 3.35 – Варіанти кодерів та кодових комбінацій для завдання 3.369

№ варіанту	Тип кодеру	Інформаційне слово **m**	Початковий розряд n_1	Кінцевий розряд n_2
1.	Рис. 3.115, а	1001	7	10
2.	Рис. 3.115, б	1011	5	8
3.	Рис. 3.115, в	1101	6	9
4.	Рис. 3.115, г	10010	8	12
5.	Рис. 3.115, д	11010	9	14
6.	Рис. 3.115, е	11011	7	10

Таблиця 3.35 – Варіанти кодерів та кодових комбінацій для завдання 3.369 (продовження)

№ варіанту	Тип кодеру	Інформаційне слово **m**	Початковий розряд n_1	Кінцевий розряд n_2
7.	Рис. 3.115, а	1101	3	7
8.	Рис. 3.115, б	1110	4	6
9.	Рис. 3.115, в	1001	5	8
10.	Рис. 3.115, г	10110	7	10
11.	Рис. 3.115, д	10010	6	8
12.	Рис. 3.115, е	10011	11	14
13.	Рис. 3.115, а	0101	4	9
14.	Рис. 3.115, б	1000	3	6
15.	Рис. 3.115, в	1001	8	11
16.	Рис. 3.115, г	11110	8	13
17.	Рис. 3.115, д	10110	6	9
18.	Рис. 3.115, е	10101	3	6

370. Перевірити результати формування кодових комбінацій згорткового коду та виправлення помилок у них, отримані в завданні 369, з використанням комп'ютерної програми, наведеної у додатку П. Які модифікації необхідно ввести в програму, наведену у додатку П, для виконання цього завдання? Обґрунтуйте свою відповідь.

371. Яка головна ідея теорії завадостійкого кодування є базовою для побудови турбокодів? Поясніть, як використовується ця ідея саме в цьому аспекті.

372. У чому полягає основа принципу правдоподібності. Наведіть власні приклади використання цього принципу в теорії завадостійкого кодування.

373. Поясніть співвідношення (3.474).

374. Що являє собою логарифм відношення функції правдоподібності?

Наведіть власні приклади використання цього параметру у теорії завадостійкого кодування.

375. Поясніть співвідношення (3.475).

376. У чому полягає головна сутність принципу побудови турбокодів? Поясніть свою відповідь на прикладі побудови турбокоду.

377. Поясніть співвідношення (3.476).

378. Що являє собою алгебра логарифму функції правдоподібності? Наведіть власні приклади виконання алгебраїчних дій над значеннями функції правдоподібності.

379. Поясніть співвідношення (3.477).

380. Що являє собою алгоритм ітеративного декодування турбокодів? Наведіть власні приклади використання цього алгоритму.

381. Поясніть структуру турбокоду, наведену на рис. 3.111.

382. Поясніть співвідношення (3.478) – (3.480).

383. Поясніть блок-схему алгоритму, яка наведена на рис. 3.112.

384. Поясніть співвідношення (3.480) – (3.486).

385. Поясніть приклад 3.71.

386. Поясніть співвідношення (3.487) – (3.494).

387. Складіть ітеративний турбокод, якщо цифрові дані, які передаються за його допомогою, мають наступні параметри:

а) $d_1 = 1, d_2 = 1, d_3 = 0, d_4 = 1, d_5 = 0, d_6 = 1, d_7 = 0, d_8 = 1, d_9 = 1$;

б) $d_1 = 0, d_2 = 0, d_3 = 1, d_4 = 1, d_5 = 0, d_6 = 0, d_7 = 1, d_8 = 1, d_9 = 0$;

в) $d_1 = 1, d_2 = 0, d_3 = 0, d_4 = 1, d_5 = 1, d_6 = 0, d_7 = 0, d_8 = 0, d_9 = 1$;

г) $d_1 = 1, d_2 = 1, d_3 = 1, d_4 = 0, d_5 = 0, d_6 = 1, d_7 = 1, d_8 = 0, d_9 = 0$;

д) $d_1 = 1, d_2 = 0, d_3 = 1, d_4 = 0, d_5 = 1, d_6 = 1, d_7 = 0, d_8 = 1, d_9 = 1$;

е) $d_1 = 0, d_2 = 1, d_3 = 0, d_4 = 1, d_5 = 0, d_6 = 0, d_7 = 1, d_8 = 0, d_9 = 1$;

ж) $d_1 = 1, d_2 = 1, d_3 = 0, d_4 = 0, d_5 = 1, d_6 = 0, d_7 = 1, d_8 = 0, d_9 = 1$;

з) $d_1 = 0, d_2 = 1, d_3 = 1, d_4 = 0, d_5 = 0, d_6 = 1, d_7 = 0, d_8 = 1, d_9 = 1$;

і) $d_1 = 1, d_2 = 1, d_3 = 0, d_4 = 0, d_5 = 1, d_6 = 1, d_7 = 0, d_8 = 0, d_9 = 1$;

к) $d_1 = 1, d_2 = 0, d_3 = 0, d_4 = 0, d_5 = 1, d_6 = 0, d_7 = 0, d_8 = 0, d_9 = 1$.

388. У чому полягає сутність створення турбокодів на основі згорткових кодів? Наведіть власні приклади формування таких турбокодів.

389. Поясніть співвідношення (3.495), (3.496).

390. Поясніть рис. 3.113.

391. Поясніть рис. 3.114.

392. Проведіть порівняльний аналіз групових, ітеративних, згорткових кодів та турбокодів. Свою відповідь обґрунтуйте та наведіть власні приклади ефективного використання таких типів кодів у сучасній цифровій електронній апаратурі.

393. У яких стандартах сучасних комп'ютерних мереж, цифрових систем обробки інформації та телекомунікаційних мереж використовуються відповідні методи завадостійкого кодування?

Перелік посилань

1. Денбновецький С.В., Мельник І.В., Писаренко Л.Д. Кодування сигналів в електронних системах. Частина 1. Параметри сигналів та каналів зв'язку та методи їхнього оцінювання. Навчальний посібник для студентів-бакалаврів напрямку підготовки 6.0508 «Електроніка» – К.: Кафедра. – 524 с.

2. Кузьмин И.В., Кедрус В.А. Основы теории информации и кодирования. – К.: Вища школа, 1977. – 238 с.

3. Пеннин П.И., Филиппов Л.И. Радиотехнические системы передачи информации. – М.: Радио и связь, 1984. – 256 с.

4. Цымбыл В.П. Основы теории информации и кодирования. – К.: Вища школа, 1992. – 294 с.

5. Дмитриев В.И. Прикладная теория информации. – М.: Высшая школа, 1989. – 320 с.

6. Панин В.В. Основы теории информации. – М.: БИНОМ, 2007. – 436 с.

7. Шеннон К. Работы по теории информации и кибернетике. – М: Издательство иностранной литературы, 1963. – 830 с.

8. Денбновецький С.В., Лещишин О.В. Електронні системи: навчальний посібник. – К.: НТУУ „КПІ", 2011. – 288 с.

9. Ширяев А.Н. Вероятность. Учебное пособие для студентов вузов. – М.: Наука, 1989. – 640 с.

10. Орлов В.А., Филиппов Л.И. Теория информации в упражнениях и задачах. – М.: Высшая школа, 1976. – 136 с.

11. Самарский А.А., Гулин А.В. Численные методы. – М.: Наука, 1989. – 432 с.

12. Денбновецкий С.В., Писаренко Л.Д., Резниченко В.К. Основы автоматизированного проектирования электронных приборов. К.: Выща школа, 1987. – 335 с.

13. Мельник І.В. Система науково-технічних розрахунків MatLab та її використання для розв'язання задач із електроніки: навчальний посібник у 2-

х томах. Т. 1. Основи роботи та функції системи. – К.: Університет «Україна», 2009. – 507 с.

14. Мельник І.В. Система науково-технічних розрахунків MatLab та її використання для розв'язання задач із електроніки: навчальний посібник у 2-х томах. Т. 2. Основи програмування та розв'язання прикладних задач. – К.: Університет «Україна», 2009. – 327 с.

15. Лунтовський А.О., Мельник І.В. Проектування та дослідження комп'ютерних мереж: навчальний посібник. – К.: Університет «Україна», 2010. – 361 с.

16. Лунтовський А.О., Мельник І.В. Основи проектування безпроводових комп'ютерних мереж: навч. посібник. – К.: Освіта України, 2011. – 362 с.

17. Мельник І.В. Інформаційні комп'ютерні мережі: Навчальний посібник для дистанційного навчання. – К.: Університет «Україна», 2006. – 250 с.

18. Мельник И.В., Гадимов Г.И. Основы построения компьютерных сетей. – Баку, Издательство «Элм», 2013. – 480 с.

19. Компьютерные сети. Принципы, технологи, протоколы. Учебник для ВУЗов. / В.Г. Олифер, Н.А. Олифер. – СПб.: Питер, 2004. – 864 с.

20. Таненбаум Э. Компьютерные сети. – BHV, СПб, 2003. – 992 с.

21. Лунтовський А.О. Технології розподілених програмних додатків. Монографія. К.: Державний університет інформаційно-комунікаційних технологій ДУІКТ, 2010. – 452 с.

22. Лунтовський А.О., Климаш М.М., Семенко А.І. Розподілені сервіси телекомунікаційних мереж та повсякденний комп'ютінг і Cloud-технології. Монографія. – Львів: Національний університет «Львівська політехніка», 2012. – 368 с.

23. Лунтовський А.О., Климаш М.М. Інформаційна безпека розподілених систем. Монографія. – Львів: Національний університет «Львівська політехніка», 2014. – 444 с.

24. Бабак В.П., Наритник Т.М., Куц Ю.В., Казмиренко В.Я. Обробка сигналів у радіоканалах цифрових систем передавання інформації. / За загальною редакцією член.-кор. НАН України В.П. Бабака. – К.: Книжкове видавництво НАНУ, 2005. – 476 с.

25. Сигорский В.П. Математический аппарат инженера. – К.: Тэхника, 1977. – 768 с.

26. Денисенко А.Н. Сигналы. Теоретическая радиотехника. Справочное пособие. – М.: Горячая линия – Телеком, 2006. – 704 с.

27. Фигурин В.А., Оболонкин В.В. Теория вероятностей и математическая статистика. Учебное пособие. – Минск: ООО «Новое знание», 2000. – 208 с.

28. Ипатов В. Широкополосные системы и кодовое разделение сигналов. Принципы и приложения. – М.: Техносфера, 2007. – 488 с.

29. Вишневский В.М., Ляхов А.И., Портной С.Л., Шахнович И.В. Широкополосные беспроводные сети передачи информации. – М.: Техносфера, 2005. – 592 с.

30. Ирвин Дж., Харль Д. Передача данных в сетях: инженерный подход: Перевод с английского. – СПБ: БХВ, 2003. – 448 с.

31. Тарасенко В.П., Маламан А.Ю., Черниченко Ю.П., Корнійчук В.І. Надійність комп'ютерних систем. – К: Корнійчук, 2007. – 256 с.

32. Домбругов Р.М. Телевидение. – К.: Вища школа, 1998. – 214 с.

33. Скляр Б. Цифровая связь. Теоретические основы и практическое применение. – М.: Издательский дом «Вильямс», 2003. – 1104 с.

34. Багатоканальний зв'язок та телекомунікаційні технології: Підручник для студентів вищих навчальних закладів / За редакцією Поповського В.В. – Харків: Компаня «Сміт», 2003. – 512 с.

35. Гельдман М.М. Аналого-цифровые преобразователи для информационно-измерительных систем. - М.: Изд-во стандартов. 1989. – 320 с.

36. Федорков Б.Г., Телец В.А. "Микросхемы ЦАП и АЦП: функционирование, параметры, применение". – М.: Энергоатомиздат, 1990. – 320 с.

37. Тимченко О.В. Методи різницевого кодування форми сигналів в сис-

темах передачі мовної інформації. – Львів: Видавництво Української академії друкарства, 2006. – 320 с.

38. Банкет В.Л., Иващенко П.В., Геер А.Э. Цифровые методы передачи информации в спутниковых системах связи. Учебное пособие. – Одесса, УГАС, 1996. – 180 с.

39. Джордейн Р. Справочник программиста персональных компьютеров типа IBM PC, XT и AT. – М.: Финансы и статистика, 1992. – 544 с.

40. Абель П. Язык ассемблера для IBM PC и программирования. – М.: Высшая школа, 1992. - 447 с.

41. http://www.cisco.com/

42. http://www.dsplib.ru/content/bpsk/bpsk.html

43. Солодов А.В. Теория информации и её применение в задачах оптимального контроля. – М.: Наука, 1967. – 534 с.

44. Каган Б.М. Электронные вычислительные машины и системы. Учебное пособие для вузов. – М.: Энергоатомиздат, 1991. – 592 с.

45. Корнев В.В. Параллельные вычислительные системы. – М.: Нолидж, 1999. – 320 с.

46. Корнейчук В.И., Тарасенко В.П. Основы компьютерной арифметики. – Киев, «Корнійчук», 2003. – 176 с.

47. Столингс В. Беспроводные линии связи и сети. – М.: Издательский дом «Вильямс», 2003. – 640 с.

48. Денбновецький С.В., Мельник І.В., Писаренко Л.Д. Кодування сигналів в електронних системах. Частина 2. Математичні основи теорії кодування. Навчальний посібник для студентів-бакалаврів напрямку підготовки 6.0508 «Електроніка». Том 1. – К.: Кафедра, 2018. – 724 с.

49. Денбновецький С.В., Мельник І.В., Писаренко Л.Д. Кодування сигналів в електронних системах. Частина 2. Математичні основи теорії кодування. Навчальний посібник для студентів-бакалаврів напрямку підготовки 6.0508 «Електроніка». Том 2. – К.: Кафедра, 2018. – 432 с.

50. Денбновецький С.В., Мельник І.В., Писаренко Л.Д. Кодування

сигналів в електронних системах. Частина 2. Математичні основи теорії кодування. Навчальний посібник для студентів-бакалаврів напрямку підготовки 6.0508 «Електроніка». Том 3. – К.: Кафедра, 2018. – 368 с.

51. Бронштейн И.Н., Семендяев К.А. Справочник по математике. Для инженеров и учащихся втузов. – М.: «Наука», Главная редакция физико-математической литературы, 1986. – 723 с.

52. Берлекэмп Э. Алгебраическая теория кодирования. – М.: Мир, 1971. – 480 с.

53. Биркгоф Г., Барти Т. Современная прикладная алгебра. – М.: Мир, 1976. – 400 с.

54. Андерсон Дж. Дискретная математика и комбинаторика. – М.: Издательский дом «Вильямс», 2004. – 960 с.

55. Вступ до алгебраїчної теорії перешкодостійкого кодування. / Волкович С.Л., Геранін В.О., Мовчан Т.В., Писаренко Л.Д., Розарінов Г.М., Хотяїнцев С.М.: Науково-методичне видання. – Київ, ВПФ УкрІНТЕІ, 2002. – 236 с.

56. Олексенко П.Ф., Коваль В.В., Розарінов Г.М., Сукач Г.О. Теоретичні основи завадостійкого кодування. Частина І. Підручник для вищих навчальних закладів за редакцією академіка НАН України В.Ф. Мачуліна. – К.: Наукова думка, 2010. – 192 с.

57. Олексенко П.Ф., Коваль В.В., Розарінов Г.М., Сукач Г.О. Теоретичні основи завадостійкого кодування. Частина ІІ. Підручник для вищих навчальних закладів за редакцією академіка НАН України В.Ф. Мачуліна. – К.: Наукова думка, 20а12. – 212 с.

58. Основи теорії телекомунікацій: підручник / О.В. Корнейко, О.В. Кувшинов, О.П. Лежнюк [та ін.]; за заг. ред. М.Ю. Ільченка. – К.: Видавництво ІСЗЗІ НТУУ «КПІ», 2010. – 786 с.

59. Кнут Д. Искусство программирования. Том 3. Сортировка и поиск. – М.: «Вильямс», 2007. — 824 с.

60. Варакин Л.Е. Системы связи с шумоподобными сигналами. – М.:

радио и связь, 1985. – 380 с.

61. https://ua-mova.livejournal.com/885374.html#/885374.html#

62. Питерсон У., Уэлдон Э. Коды, исправляющие ошибки. – М.: Советское радио, 1986. – 593 с.

63. Блейхут Р. Теория и практика кодов, контролирующих ошибки: пер. с англ. – М.: Мир, 1986. – 576 с.

64. Фирсов В.А. Теория информации. – Самара, Издательство самарского государственного аэрокосмического университета, 2011. – 128 с.

65. Прохоров В.С. Теория информации. Лекции. – Електронний ресурс. Режим доступу: http://profbeckman.narod.ru/Informat.files/Teorinf.pdf

66. Сукачев Э.А., Бухан Д.Ю. Корреляционный анализ детерминированных сигналов. Компьютеризированый курс. Монография. – Одесса, 2014. – 134 с.

67. Бородин Л.Ф. Введение в теорию помехоустойчивого кодирования. – М.: Советское радио, 1968. – 408 с.

68. Теория электрической связи: учебное пособие / К.К. Васильев, В.А. Глушков, А.В. Дормидонтов, А.Г. Нестеренко. Под общ. ред. К.К. Васильева. – Ульяновск: УлГТУ, 2008. – 452 с.

69. Финк Л.М. Теория передачи дискретных сигналов. – М.: "Советское радио", 1970. – 728 с.

70. Вернер М. Основы кодирования. — М.: Техносфера, 2004. – 288 с.

70. http://informkod.narod.ru/5_5item.htm

71. http://yourtutor.narod.ru/cyclic/CyclicCodes.htm

72. http://dvo.sut.ru/libr/opds/i285vino/2.htm

73. http://book.itep.ru/2/28/corec_28.htm

74. http://stor.boom.ru/uch/din/1/15/153.htm

75. Мельник І.В. Лабораторний парктикум з курсу «Проектування інформаційних електронних систем та мереж». Електронний навчальний посібник для студенів-спеціалістів V курсу факультета електроніки спеціальності електронні прилади та пристрої. – 166 с.

76. Дьяконов В.П. Simulink 5/6/7: Самоучитель. – М.: ДМК – Пресс. – 2008. – 784 с.

77. Дьяконов В.П., Пеньков А.А. MatLab и Simulink в электроэнергетике. Справочник. – М.: Горячая линия – Телеком. – 2009. – 816 с.

78. Прикладная теория цифровых автоматов / К. Г. Самофалов, А. М. Ромлинкевич, В. Н. Валуйский, Ю. С. Каневский, М. М. Пиневич.— К.: Вища шк. Головное изд-во, 1987. — 375 с.

79. Королёв А.И. Коды и устройства помехоустойчивого кодирования информации. – Минск, Бестпринт, 2002. – 238 с.

80. https://ru.wikipedia.org/wiki/Код_Боуза_—_Чоудхури_—_Хоквингема

81. Lathi B.P. Modern digital and analog communication systems. – Holt, Rinehart and Whinston Inc., 1989. – 720 p.

82. http://www.nav0911.narod.ru/rs.doc&rct=j&q=&esrc=s&sa=U&ved=0ahUKEwjO-a7m5fnbAhUB2ywKHY6QDzEQFggxMAU&usg=AOvVaw3mA-aP2Lj0JS1QTE94g8ZT

83. Поляков А.К. Языки VHDL и Verilog в проектировании цифровой аппаратуры. — М.: СОЛОН-Пресс, 2003. – 320 с.

84. Основи теорії телекомунікацій: підручник / О.В. Корнейко, О.В. Кувшинов, О.П. Лежнюк [та ін.]; за заг. ред. М.Ю. Ільченка. – К.: Видавництво ІСЗЗІ НТУУ «КПІ», 2010. – 786 с.

85. http://www.nasonline.org/publications/biographical-memoirs/memoir-pdfs/elias-peter.pdf

86. Никитин Г. И. Сверточные коды: Учеб. пособие. Санкт-Петербургский государственный университет аэрокосмического приборостроения. – СПб., 2001. – 80 с. http://techlibrary.ru/b1/2v1j1l1j1t1j1o_2k.2q._2z1c1g1r1t1p1y1o2c1f_1l1p1e2c._2001.pdf

ДОДАТОК Л. Програма для формування коду Файра та декодування його кодових послідовностей

```
%Fuction for calculation the sequences of Fire Codes
%and for its decoding.
%The program formed Fire code with parameters (15,35)
%and can find at least 3 errors.
%Input parameters:
%vin - binary sequence. Only value 0 and 1 for vector
vin is allows. Number of
%bites of coded sequence is from 8 to 15, and for
decoded sequence - from 17 to 31.
%Another function parameters are:
%nop - number of necessary operation.
%Possible values of this parameter:
%1 - forming of Fire code.
%In this case output vector is coded binary sequence.
%2 - decoding of Fire code sequence.
%Nerr - number of packet errors, which can be detected.
%Usually Nerr=3.
%In this case output vector is decoded binary sequence.
Error positions are also noted.
%Examples of using this function
%fr=Fire([1,0,1,0,0,1,0,1],1,3)
%fr =
%  1  0  1  1  1  1  1  0  0  0  1  0  1  1  1  1  1
%>> dfr=Fire(fr,2,3)
%No errors. Coded sequence is 1 0 1 0 0 1 0 1
%dfr =
%   1   0   1   0   0   1   0   1
```

```
%>> Errsind=[1,0,1];
%>> fr(5:7)=xor(fr(5:7),Errsind);
%>> fr
%fr =
%  1  0  1  1  0  1  0  0  0  0  1  0  1  1  1  1  1
%>> dfr=Fire(fr,2,3);
%Error syndrome: S=1  0  1, found in input Fire code at
the w=5-th position.
%Correct Fire code is:
%FC=1  0  1  1  1  1  1  0  0  0  1  0  1  1  1  1  1
%Coded sequence is: c=1  0  1  0  1  1  1  0
function Vout=Fire(vin,nop,Nerr)
    nv=length(vin);
    dd1='Wrong input vector! Bits of vector have t';
    dd2='o be equal 0 or 1. Check input data!';
    ddstr=[dd1,dd2];
    for ii=1:nv
     if(~((vin(ii)==1)|(vin(ii)==0))) error (ddstr); end;
    end; %for ii=1:nv
    rtr1='Wrong number of operation nop. This valu';
    rtr2='e can be equal 1 for coding task or 2 f';
    rtr3='or decoding task'; ddstr2=[rtr1,rtr2,rtr3];
    if(~((nop==1)|(nop==2))) error (ddstr2); end;
    etr1='Wrong amount of bites in input vector. Coded sequenc';
   etr2='e can inclued not less then 8 bites but not mor';
    etr3='e then 11 bytes'; estr=[etr1,etr2,etr3];
    if (((nv<8)|(nv>11))&(nop==1)) error (estr); end;
    ttr1='Wrong amount of bites in input vecto'
    ttr23='r. Decoded sequenc';
    ttr3='e can inclued not less then 17 bites but not mor';
```

```
ttr4='e then 31 bytes'; tstr=[ttr1,ttr2,ttr3, ttr4];
if ((nv<17)|(nv>31))&(nop==2) error (tstr); end;
TP1=[1,0,0,1,1];
if (Nerr==3) TP2=[1,0,0,0,0,1]; end;
if(nop==1)
    Vout1=CRC(vin,1,1);
    Vout=Bin_Product(Vout1,TP2);
end; %if (nop==1)
if(nop==2)
    RES1M=Bin_Division(vin,TP1); RES1=RES1M(2,:);
    RES2M=Bin_Division(vin,TP2); RES2=RES2M(2,:);
    nr1=length(RES1); Z1=zeros(1,nr1);
    nr2=length(RES2); Z2=zeros(1,nr2);
    if (RES1==Z1)
        if (RES2==Z2)
            wr1='No errors found. Coded sequence is ';
            VoutM=Bin_Division(RES1M(1,:),TP2);
            Vout=VoutM(1,:);  wr2=num2str(Vout);
            wrt=strcat(wr1,': c=',wr2); disp(wrt);
        end; %if (RES2==Z2)
    end; %if (RES1==Z1)
    cp1=(RES1~=Z1); cpr1=sum(cp1);
    cp2=(RES2~=Z2); cpr2=sum(cp2);
    if (cpr1~=0)|(cpr2~=0)
        for ii=1:nr1-1
          if ((RES1(ii)==0)&(RES1(ii+1)==1)) break; end;
        end; %for ii=1:nr1-1
        RESSHORT1=RES1(ii+1:nr1);
        for jj=1:nr2-1
            if ((RES2(jj)==0)&(RES2(jj+1)==1)) break; end;
```

```
end; % for jj=1:nr2-1
RESSHORT2=RES2(jj+1:nr2);
nrsh1=length(RESSHORT1);
nrsh2=length(RESSHORT2);
RESCOMP2=RESSHORT2; RESCOMP1=RESSHORT1;
if (nrsh1>nrsh2)
    nz=nrsh1-nrsh2; ZRES=zeros(1,nz);
    RESCOMP2=[ZRES,RESSHORT2];
end;%if (nrsh1>nrsh2)
if (nrsh2>nrsh1)
    nz=nrsh2-nrsh1; ZRES=zeros(1,nz);
    RESCOMP1=[ZRES,RESSHORT1];
end;%if (nrsh2>nrsh1)
FC=vin; ww=1;
cpres=(RESCOMP1~=RESCOMP2);
crer_cmp=sum(cpres);
while (crer_cmp~=0)&(ww<nv+1)
    FC=[FC,0];
    RES1M_NEW=Bin_Division(FC,TP1);
    RES1_NEW=RES1M_NEW(2,:);
    RES2M_NEW=Bin_Division(FC,TP2);
    RES2_NEW=RES2M_NEW(2,:);
    nrsh1_new=length(RES1_NEW);
    nrsh2_new=length(RES2_NEW);
    RESCOMP1=RES1_NEW; RESCOMP2=RES2_NEW;
    if (nrsh1_new>nrsh2_new)
        nz_new=nrsh1_new-nrsh2_new;
        ZRES=zeros(1,nz_new);
        RESCOMP2=[ZRES,RES2_NEW];
        SIND_LONG=RES1_NEW;
```

```
    end; %if (nrsh1_new>nrsh2_new)
    if (nrsh2_new>nrsh1_new)
        nz_new=nrsh2_new-nrsh1_new;
        ZRES_NEW=zeros(1,nz_new);
        RESCOMP1=[ZRES_NEW,RES1_NEW];
        SIND_LONG=RES2_NEW;
    end; %if (nrsh2_new>nrsh1_new)
    cpres=(RESCOMP1~=RESCOMP2);
    crer_cmp=sum(cpres); ww=ww+1;
end; %while (RESCOMP1~=RESCOMP2)&(ww<nv+1)
nsind=length(SIND_LONG);
if (nsind>length(RES1_NEW))
    cd=nsind-length(RES1_NEW);
    LZ=zeros(1,cd); RES1_NEW=[LZ,RES1_NEW];
end;%if (nsind>length(RES1_NEW))
if (nsind>length(RES2_NEW))
    cd=nsind-length(RES2_NEW);
    LZ=zeros(1,cd); RES2_NEW=[LZ,RES2_NEW];
end;%if (nsind>length(RES2_NEW))
cont=(SIND_LONG~=zeros(1,nsind));
contsum=sum(cont);
cont1=(RES1_NEW~=zeros(1,nsind));
contsum1=sum(cont1);
cont2=(RES2_NEW~=zeros(1,nsind));
contsum2=sum(cont2);
if(contsum==0)&(contsum1~=0)
     SIND_LONG=RES1_NEW;
end; % if(contsum==0)&(contsum1~=0)
if(contsum==0)&(contsum2~=0)
     SIND_LONG=RES2_NEW;
```

```
end; % if(contsum==0)&(contsum2~=0)
for jk=1:nsind
    if((SIND_LONG(jk)==0)&(SIND_...
        LONG(jk+1)==1)) break; end;
end;%for jk=1:nsind
SIND=SIND_LONG(jk+1:nsind);
if (ww<nv+1)
    lerr=length(SIND);
    VCORR=xor(vin(ww-1:ww+lerr-2),SIND);
    FCCorr=[vin(1:ww-2),VCORR,...
         vin(ww+lerr-1:nv)];
    wrte1='Error syndrome: S=';
    wrte2=num2str(SIND);
    wrte3=', found in input Fire code at the w= ';
    if (lerr==Nerr)   NPos=ww-1; end;
    if (lerr==Nerr-1) NPos=ww; end;
    if (lerr==Nerr-2) NPos=ww-1; end;
    wrte4=num2str(NPos);
    wrte5='-th position. ';
    wrt1=strcat(wrte1,wrte2,wrte3,...
         wrte4,wrte5);
    disp(wrt1);
    wrte6='Correct Fire code is:';
    wrte7='FC=';
    wrte8=num2str(FCCorr);
    disp(wrte6);
    wrt2=strcat(wrte7,wrte8);
    disp(wrt2);
    wrte9='Coded sequence is: c=';
    R1M=Bin_Division(vin,TP1);
```

```
                VM=Bin_Division(R1M(1,:),TP2);
                Vout=VM(1,:); wrte10=num2str(Vout);
                wrt3=strcat(wrte9,wrte10);
                disp(wrt3);
            end;%if (ww<nv+1)
            if (ww>=nv+1)
                wrte1='Wrong Fire code sequence. Error ';
                wrte2='syndrome S not foun';
                wrte3='d. Please check input data.';
                Vout=[];
                wrte=strcat(wrte1,wrte2,wrte3);
                disp(wrte);
            end;%if (ww>=nv+1)
        end; %if(cpr1~=0)|(cpr2~=0)
    end; %if(nop==2)
return
```

Результати тестування програми

```
>> fr=Fire([1,0,1,0,0,1,0,1],1,3);
>> dfr=Fire(fr,2,3);
No errors found. Coded sequence is: c=1 0 1 0 0 1 0 1
>> Errsind=[1,0,1];
>> fr(5:7)=xor(fr(5:7),Errsind);
>> dfr=Fire(fr,2,3);
Error syndrome: S=1  0  1, found in input Fire code at
the w=5-th position.
Correct Fire code is:
FC=1  0  1  1  1  1  1  0  0  0  1  0  1  1  1  1  1
Coded sequence is: c=1  0  1  0  0  1  0  1
>> fr=Fire([1,0,0,0,0,1,0,1],1,3);
```

```
>> dfr=Fire(fr,2,3);
No errors found. Coded sequence is: c=1 0 0 0 0 1 0 1
>> Errsind=[0,1,1];
>> fr(12:14)=xor(fr(12:14),Errsind);
>> dfr=Fire(fr,2,3);
Error syndrome: S=1  1, found in input Fire code at the
w=13-th position.
Correct Fire code is:
FC=1  0  0  1  1  0  0  1  0  0  0  0  1  0  1  1  1
Coded sequence is: c=1  0  0  0  0  1  0  1
>> fr=Fire([1,1,1,0,0,1,0,1],1,3);
>> dfr=Fire(fr,2,3);
No errors found. Coded sequence is: c=1 1 1 0 0 1 0 1
>> Errsind=[1,0,1,1,1];
>> fr(10:14)=xor(fr(10:14),Errsind);
>> dfr=Fire(fr,2,3);
Wrong Fire code sequence. Error syndrome S not found.
Please check input data.
>>
```

ДОДАТОК М. Програма для формування систематичних кодів Боуза – Чоудхурі – Хоквінгема (15, 7) та (15, 5) та декодування їхніх кодових послідовностей

```
%Fuction for calculation the sequences of SCh Codes and
%for decoding of these codes. Input parameters:
%vin - binary sequence. Only value 0 and 1 for vector
%vin is allows. Number of bites of coded sequence is
%from 1 to 7 for code BCH (15,7) or from 1 to 5 for
%code BCH (15,5). Number of bites of coded sequence is
%usually 15.
%Another function parameters are:
%nop - number of necessary operation.
%Possible values of this parameter:
%1 - forming of code BCH (15,7) for double-errors
%correction or code BCH (15,5) for tripple-errors
%correction.
%In this case output vector is coded coded binary
sequence.
%2 - decoding BCH sequence.
%In this case output vector is decoded binary sequence.
%Error position is also noted.
%TOC - type of BCH code.
%Possible values of this parameter:
%1 - code BCH (15,7) for correction of double errors.
%2 - code BCH (15,5) for correction of tripple-errors.
%Examples of using this function
%>> c=BCH([1,1,1,0,0,1,1],1,1)
%c =
%   1  1  1  0  0  1  1  0  0  0  0  0  1  0  0
```

```
%>> d=BCH(c,2,1)
%BCH code is correct. Coded sequence is vout=1 1 1 0 0 1 1
    %d =
    %   1   1   1   0   0   1   1
    %>> c(14)=1;
    %>> d=BCH(c,2,1)
%BCH code is incorrect. Errors in the 2-th digit detected.
%Corrected code sequence is:
%vin=1  1  1  0  0  1  1  0  0  0  0  0  1  0  0.
%Coded sequence is vout=1  1  1  0  0  1  1
    %d =
    %   1   1   1   0   0   1   1
    %>> c(13)=0; c(14)=1;
    %>> d=BCH(c,2,1)
%BCH code is incorrect. Errors in the 3-th and 2-th
%digit detected.
%Corrected code sequence is:
%vin=1  1  1  0  0  1  1  0  0  0  0  0  1  0  0.
%Coded sequence is vout=1  1  1  0  0  1  1
    %d =
    %   1   1   1   0   0   1   1
    %>>
    %>> c=BCH([1,1,1,0,0],1,2)
    %c =
    %  1  1  1  0  0  0  0  1  0  1  0  0  1  1  0
    %>> c(4)=1; c(7)=1; c(9)=1;
    %>> c=BCH(c,2,2)
%BCH code is incorrect. Errors in the 12-th 9-th
%and 7-th digit detected.
%Corrected code sequence is:
```

```
%vin=1  1  1  0  0  0  0  1  0  1  0  0  1  1  0.
%Corrected code sequence is vin=1  1  1  0  0
    %c =
    %     1     1     1     0     0
    %>>
%See also: CRC, Bin_Division and Bin_Product
    function Vout=BCH(vin,nop,TOC)
        nv=length(vin);
        dd1='Wrong input vector! Bits of vector have t';
        dd2='o be equal 0 or 1. Check input data!';
        ddstr=[dd1,dd2];
        for ii=1:nv
            if(~((vin(ii)==1)|(vin(ii)==0))) error ...
                (ddstr); end;
        end; %for ii=1:nv
        rtr1='Wrong number of operation nop. This valu';
        rtr2='e can be equal 1 for coding task or 2 f';
        rtr3='or decoding task';
        ddstr2=[rtr1,rtr2,rtr3];
        if(~((nop==1)|(nop==2))) error (ddstr2); end;
        ftr1='Wrong number of type of code TOC. This valu';
        ftr2='e can be equal 1 for code BCH (15,7) or 2 f';
        ftr3='for code BCH (15,5).';
        fddstr2=[ftr1,ftr2,ftr3];
        if(~((TOC==1)|(TOC==2))) error (fddstr2); end;
        etr1='Wrong amount of bites in ...
                input vector. Coded sequenc';
        etr2='e can inclued not less then ...
                2 bites but not mor';
        etr3='e then 7 bytes. 6 and 7 ...
```

```
     bites only for code (15,7) is possible';
 estr=[etr1,etr2,etr3];
if (((nv<2)|(nv>7))&(nop==1)&(TOC==1)) error (estr); end;
 if (((nv<2)|(nv>5))&(nop==1)&(TOC==2)) error (estr); end;
ttr1='Wrong amount of bites in i...
     nput vector. Decoded sequenc';
 ttr2='e can inclued only 15 bites';
 tstr=[ttr1,ttr2];
 if ((nv~=15)&(nop==2)) error (tstr); end;
 if (TOC==1) TP=[1,1,1,0,1,0,0,0,1]; end;
 if (TOC==2) TP=[1,0,1,0,0,1,1,0,1,1,1]; end;
 if ((nv==2)&(nop==1)&(TOC==1))...
       VS=[0,0,0,0,0,vin,0,0,0,0,0,0,0,0]; end;
 if ((nv==3)&(nop==1)&(TOC==1))...
     VS=[0,0,0,0,vin,0,0,0,0,0,0,0,0]; end;
 if ((nv==4)&(nop==1)&(TOC==1))...
     VS=[0,0,0,vin,0,0,0,0,0,0,0,0]; end;
 if ((nv==5)&(nop==1)&(TOC==1))...
     VS=[0,0,vin,0,0,0,0,0,0,0,0]; end;
 if ((nv==6)&(nop==1)&(TOC==1))...
     VS=[0,vin,0,0,0,0,0,0,0,0]; end;
 if ((nv==7)&(nop==1)&(TOC==1))...
     VS=[vin,0,0,0,0,0,0,0,0]; end;
 if ((nv==2)&(nop==1)&(TOC==2))
     VS=[0,0,0,vin,0,0,0,0,0,0,0,0,0]; end;
 if ((nv==3)&(nop==1)&(TOC==2))...
     VS=[0,0,vin,0,0,0,0,0,0,0,0,0,0]; end;
 if ((nv==4)&(nop==1)&(TOC==2))...
       VS=[0,vin,0,0,0,0,0,0,0,0,0,0]; end;
 if ((nv==5)&(nop==1)&(TOC==2))...
```

```
      VS=[vin,0,0,0,0,0,0,0,0,0]; end;
if (nop==1)
    MC=Bin_Division(VS,TP); RC=MC(2,:);
     NVS=length(VS); NRC=length(RC);
    DIFL=NVS-NRC; ZL=zeros(1,DIFL);
     RCM=[ZL,RC]; Vout=xor(RCM,VS);
end;
OUTEXT='BCH code is correct. Coded sequence is vout=';
ERROUTTEXT1='BCH code is incorrect. Errors in the ';
ERROUTTEXT3=' and ';
ERROUTTEXT4='-th digits detected';
ERROUTTEXT5='. Corrected code sequence is vin=';
ERROUTTEXT6='. Coded sequence is vout=';
if(nop==2)
    MC=Bin_Division(vin,TP);
    Res=MC(1,:); Rem=MC(2,:);
    SRem=sum(Rem);
    if (SRem==0)
        if (TOC==1) Vout=vin(1:7);
             else Vout=vin(1:5); end;
        VoutVect=num2str(Vout);
        VoutSTR=strcat(OUTEXT, VoutVect);
        disp(VoutSTR);
    end; %if (SRem==0)
if (SRem>0)
    VinShift=vin; iii=0;
    while (((SRem>2)&(TOC==1))|((SRem>3)&...
     (TOC==2))&(iii<15))
         VShift=[VinShift(2:15),VinShift(1)];
         MCS=Bin_Division(VShift,TP);
```

```
     RemNew=MCS(2,:); SRem=sum(RemNew);
     iii=iii+1;
     VinShift=VShift;
end;%while (((SRem>2)&(TOC==1))|...
if (iii>0)
    Nrem2=length(RemNew);
    NAddZr2=15-Nrem2;
     NZrVect2=zeros(1,NAddZr2);
    RemPlus=[NZrVect2,RemNew];
    CorrSeq=xor(VinShift,RemPlus);
    VShift=CorrSeq;
    for jjj=1:iii
        VinShift=[VShift(15),...
         VShift(1:14)];
        VShift=VinShift;
    end; %for jjj=1:iii
end; %if (iii>0)
VoutVectCode=num2str(VShift);
if (TOC==1) Vout=VShift(1:7);
          else Vout=VShift(1:5); end;
VoutVectDec=num2str(Vout);
if (iii==0)
    Nrem=length(Rem);
    NAddZr=15-Nrem;
     NZrVect=zeros(1,NAddZr);
    RemPlus=[NZrVect,Rem];
    CorrSeq=xor(vin,RemPlus);
    if (TOC==1) Vout=CorrSeq(1:7);
         else Vout=CorrSeq(1:5); end;
    VoutVectCode=num2str(CorrSeq);
```

```
    VoutVectDec=num2str(Vout);
end; %if (iii==0)
Pos(1)=0; Pos(2)=0; Pos(3)=0;
 SP1=0; SP2=0; SP3=0;
    for ii=1:15
        if ((RemPlus(ii)==1)&(Pos(1)==0))
            Pos(1)=ii;
            if (iii==0)
                  SP1=num2str(15-Pos(1)+1);
                else
                  Num1=15-Pos(1)+1+15-iii;
                  if (Num1>15)...
                      Num1=Num1-15; end;
                  SP1=num2str(Num1);
                end; %if (iii==0)
        end;%if ((RemPlus(ii)==1)&...
        if ((RemPlus(ii)==1)&(Pos(1)~=0)&...
              (Pos(1)~=ii)&(Pos(2)==0))
            Pos(2)=ii;
            if (iii==0)...
                  SP2=num2str(15-Pos(2)+1);
                else
                 Num2=15-Pos(2)+1+15-iii;
                 if (Num2>15) Num2=Num2-15; end;
                    SP2=num2str(Num2);
                end; %if (iii==0)
        end; %if ((RemPlus(ii)==1)&...
        if ((RemPlus(ii)==1)&(Pos(1)~=0)&...
              (Pos(1)~=ii)&(Pos(2)~=0))&...
              (Pos(2)~=ii)
```

```
                    Pos(3)=ii;
                    if (iii==0) SP3=num2str...
                          (15-Pos(3)+1);
                        else
                            Num3=15-Pos(3)+1+...
                              15-iii;
                            if (Num3>15) ...
                                  Num3=Num3-15; end
                            SP3=num2str(Num3);
                        end; %if (iii==0)
                end; %if ((RemPlus(ii)==1)&...
            end; %for ii==1:15
if (SP2==0)
     VoutSTR=strcat(ERROUTTEXT1,SP1,...
     ERROUTTEXT4,ERROUTTEXT5,VoutVectCode,...
     ERROUTTEXT6,VoutVectDec); end;
if ((SP2~=0)&(SP3==0))
     VoutSTR=strcat(ERROUTTEXT1,SP1,'-th',...
     ERROUTTEXT3,SP2,ERROUTTEXT4,...
     ERROUTTEXT5,VoutVectCode, ...
     ERROUTTEXT6,VoutVectDec); end;
         if (SP3~=0)
              VoutSTR=strcat(ERROUTTEXT1,...
              SP1,'-th ',SP2,'-th ',...
              ERROUTTEXT3,SP3,ERROUTTEXT4,...
              ERROUTTEXT5,VoutVectCode,...
              ERROUTTEXT5,VoutVectDec); end;
         disp(VoutSTR);
  end; %if (SRem>0)
end; %if(nop==2)
```

```
    return
```

Результати тестування програми

```
>> c=BCH([1,0,1,1,0],1,2)
c =
     1   0   1   1   0   0   1   0   0   0   1   1    1   1   0
>> c=BCH([1,0,1,1,0,1,0],1,1)
c =
     1   0   1   1   0   1   0   1   0   1   1   1   1   0    0
>> c=BCH(c,2,1)
BCH code is correct. Coded sequence is
vout=1   0   1   1   0   1   0
c =
     1    0    1    1    0    1    0
>> c(5)=1;
>> d=BCH(c,2,1)
BCH code is incorrect. Errors in the 11-th digit
detected. Corrected code sequence is
vin=1   0   1   1   0   1   0   1   0   1   1   1   1   0   0.
Coded sequence is vout=1   0   1   1   0   1   0
d =
     1    0    1    1    0    1    0
>> c(8)=0;
>> d=BCH(c,2,1)
BCH code is incorrect. Errors in the 11-th and 8-th
digits detected. Corrected code sequence is
vin=1   0   1   1   0   1   0   1   0   1   1   1   1   0   0.
Coded sequence is vout=1   0   1   1   0   1   0
d =
     1    0    1    1    0    1    0
>> c(11)=0;
```

```
>> d=BCH(c,2,1)

BCH code is incorrect. Errors in the2-th and14-th
digits detected. Corrected code sequence is
vin=1  1  1  1  1  1  0  0  0  1  0  1  1  1  0.
Coded sequence is vout=1  1  1  1  1  1  0
d =
     1     1     1     1     1     1     0
>> c=BCH([1,0,1],1,2)
c =
     0  0  1  0  1  0  0  1  1  0  1  1  1  0  0
>> d=BCH(c,2,2)
BCH code is correct. Coded sequence is
vout=0  0  1  0  1
d =
     0   0   1   0   1
>> c(4)=1;
>> d=BCH(c,2,2)
BCH code is incorrect. Errors in the 12-th digits
detected. Corrected code sequence is
vin=0  0  1  0  1  0  0  1  1  0  1  1  1  0  0.
Coded sequence is vout=0  0  1  0  1
d =
     0     0     1     0     1
>> c(6)=1;
>> d=BCH(c,2,2)
BCH code is incorrect. Errors in the 12-th and 10-th
digits detected. Corrected code sequence is
vin=0  0  1  0  1  0  0  1  1  0  1  1  1  0  0.
Coded sequence is
```

```
vout=0  0  1  0  1
d =
     0   0   1   0   1
>> c(7)=1;
>> d=BCH(c,2,2)
BCH code is incorrect. Errors in the 12-th 10-th and
9-th digits detected. Corrected code sequence is
vin=0  0  1  0  1  0  0  1  1  0  1  1  1  0  0.
Corrected code sequence is vin=0  0  1  0  1
d =
     0   0   1   0   1
>> c(11)=0;
>> d=BCH(c,2,2)
BCH code is incorrect. Errors in the 9-th 7-th and
2-th digits detected. Corrected code sequence is
vin=0  0  1  1  1  1  0  1  0  1  1  0  0  1  0.
Corrected code sequence is vin=0  0  1  1  1
d =
     0     0     1     1     1
>>
```

ДОДАТОК Н. Програмні модулі для реалізації алгебраїчних та поліноміальних операцій у полях Галуа GF(2^m)

Функція conv2bin (i, m) для перетворення десяткових чисел до векторів двійкових послідовностей, які їм відповідають

```
%The function for converting the vector of decimal integer
%values into matrix of bites, which presented this
%values in the binary system. The argument of this
%function have to be the v ector integer positive value.
%Result of calculation is the matrix, in which collomns
%the bites of coded values are presenteed.
%Examples of using this function:
%>> conv2bin(11,3)
%ans =
%     1
%     0
%     1
%     1
%>> conv2bin([7,4,1,2],4)
%ans =
%     0     0     0     0
%     1     1     0     0
%     1     0     0     1
%     1     0     1     0
function Mout=conv2bin(c,m)
    df1='Error of input vector. I';
    df2='t have to be included only p';
    df3='ositive integer values.';
    df=[df1,df2,df3]; lp=c-floor(c);
    if ((c<0)|(lp~=0)) error (df); end;
    nb=length(c); Mout=[];
    for nn=1:nb
        Res=c(nn); bv=[]; ResD=c(nn);
```

```
        while (ResD>1)
            if (mod(ResD,2)==0) bv=[0,bv];
                else bv=[1,bv]; end;
            ResD=floor(ResD/2);
        end; %while (ResD>1)
        if (c(nn)>0) bv=[1,bv]; else bv=[0,bv]; end;
        ndig=length(bv);
        if (ndig<m)
            df=m-ndig;
            NZ=zeros(1,df);
            bv=[NZ,bv];
        end; %if (ndig<m)
        Mout=[Mout,bv'];
    end; %for nn=1:nb
return
```

Функція convbin2dec (i) для перетворення векторів двійкових послідовностей до відповідних десяткових чисел

```
%The function for converting of vector of bites
%into the decimal value. Vector . The argument of this
%function is the vector, which elements are 0 or 1
%and have to be written as collomn.
%The results of calculation is correspondend decimal
%value.
%For transforming the set of binary values into decimal
%value, they have to be presented as matrix, in which
%the bit sequences reesnted in collons, and sets of
%values - in the rows, which corresponed to that
%collons.
%Examples of using this function:
%>> convbin2dec([1,1,0,1]')
%ans =
```

```
%    13
%>>
%>> convbin2dec([1,1,0,1]')
%ans =
%    13
%>>
%>> a=[1,1,0,1]';
%>> b=[0,1,1,0]';
%>> c=[a,b]
%c =
%     1     0
%     1     1
%     0     1
%     1     0
%>> convbin2dec(c)
%ans =
%    13     6
%>>
function dvout=convbin2dec(vin)
    NS=size(vin); nv=NS(1); ng=NS(2);
    dd1='Wrong input matrix! Bits of matrix have t';
    dd2='o be equal 0 or 1. Check input data!';
    ddstr=[dd1,dd2];
    for jj=1:ng
       for ii=1:nv
         if(~((vin(ii,jj)==1)| ...
                  (vin(ii,jj)==0))) error (ddstr); end;
       end; %for ii=1:nv
    end;    %for jj=1:ng
    dvout=[];
```

```
    for jjj=1:ng
       dv=0;
       for iii=1:nv
          p=nv-iii;
           if (vin(iii,jjj)==1) dv=dv+2.^p; end;
       end;%for iii=1:nv
       dvout=[dvout,dv];
   end; %for jjj=1:ng
return
```

Функція sumgf2m(m, i, j) для сумування елементів полів Галуа *GF*(2^m)

```
%The function for summing Galuas Field GF(2^m),
%where m - the order of the field.
%Functions parameters are:
%1. m - order of the field. Possible value of
%this parameter are 3,4,5,6,7,8,12,16.
%2. ii, jj - suming elements.
%Possible vaues of these paraemters: 0<=i<=2^m-1
%Both m, ii and jj have to be integer values.
%This function is widely used in discrete mathematic
%for forming the codes of digital sequences.
%For example, Reed - Solomon code based on arithmetic
%operations in Galoas Fields.
%Examples of using this function:
%>> sumgf2m(3,2,5)
%ans =
%     7
%>>
%>> sumgf2m(4,5,7)
```

```
%ans =
%     2
%>> sumgf2m(8,240,122)
%ans =
%   138
%>>
%>> sumgf2m(3,[4,5,7,3],[2,7,2,1])
%ans =
%     6     2     5     2
%>>
%See also:  powgf2m(m,ii), loggf2m, prodgf2m,
%           divgf2m, conv2bin, convbin2dec
function kk=sumgf2m(m,ii,jj)
    n1=length(ii); n2=length(jj);
    fd1='Wrong input papameter m, order o';
    fd2='f Galoas Field GF(2^m). This va';
    fd3='lue have to be integer value bot';
    fd4='h from 3 to 8 or 12 or 16.';
    df=[fd1,fd2,fd3,fd4];
    lpp=m-floor(m);
    if (lpp~=0) error (df); end;
    if ((m<3)|((m>8)&(m~=12)&(m~=16))) error (df); end;
    qw1='Wrong input vectors ii and jj. T';
    qw2='he vectors must have same length.';
    qw=[qw1,qw2];
    if(n1~=n2)error (qw); end;
    rt1='Wrong input papameter ii or jj, the power fu';
    rt2='nction arguments. Its should be smaller, t';
    rt3='hen maximum element of Galoas Field 2^m-';
    rt4='1, where m - order of the Field, first fu';
```

```
    rt5='nction parameter.';
    rt=[rt1,rt2,rt3,rt4,rt5];
    for ij=1:n1
        sd1=(2.^m-1)-ii(ij); sd2=(2.^m-1)-jj(ij);
        if (sd1<0) error (rt); end;
        if (sd2<0) error (rt); end;
    end;
    sh1=conv2bin(ii,m);
    sh2=conv2bin(jj,m);
    kkbin=xor(sh1,sh2);
    kk=convbin2dec(kkbin);
return;
```

Функція powgf2m(m, i) для піднесення числа 2 до значень елементів i, які належать полю Галуа *GF*(2^m)

```
%The power function 2^i for Galuas Field GF(2^m),
%where m - the order of the field.
%Functions parameters are:
%1. m - order of the field. Possible value of
%this parameter are 3,4,5,6,7,8,12,16.
%2. i - function argument.
%Possible vaues of this paraemter: -2^m-1<=i<=2^m-1.
%Both m and i have to be integer values, but negative
%values of parameter i is also possible.
%This function is widely used in discrete mathematic
%for forming the codes of digital sequences.
%For example, Reed - Solomon code based on arithmetic
%operations in Galoas Fields.
%Examples of using this function:
%>> powgf2m(3,2)
%ans =
```

```
%     4
%>> powgf2m(4,5)
%ans =
%     6
%>>
%>> powgf2m(8,240)
%ans =
%    44
%>> powgf2m(8,-5)
%ans =
%   108
%See also:  loggf2m, prodgf2m, divgf2m,
%           sumgf2m, conv2bin, convbin2dec
function pwi=powgf2m(m,ii)
    fd1='Wrong input papameter m, order o';
    fd2='f Galoas Field GF(2^m). This va';
    fd3='lue have to be integer value bot';
    fd4='h from 3 to 8 or 12 or 16.';
    df=[fd1,fd2,fd3,fd4];
    lpp=m-floor(m);
    if (lpp~=0) error (df); end;
    if ((m<3)|((m>8)&(m~=12)&(m~=16))) error (df); end;
    sd=(2.^m-1)-abs(ii);
    rt1='Wrong input papameter i, the power fu';
    rt2='nction argument. It should be smaller, t';
    rt3='hen maximum element of Galoas Field 2^m-';
    rt4='1, where m - order of the Field, first fu';
    rt5='nction parameter.';
    rt=[rt1,rt2,rt3,rt4,rt5];
    if (sd<0) error (rt); end;
    lpp2=ii-floor(ii);
```

```
qt1='Wrong input papameter i, the power fu';
qt2='nction argument. It should be integer value.';
qt=[qt1,qt2];
if (lpp2~=0) error (qt); end;
if (ii<0) ii=(2.^m-1)+ii; end;
pwi=1;
if (ii==2.^m-1) pwi=0; end;
if (ii==0) pwi=1; end;
if (m==3)  TP=[1,0,1,1]; end;
if (m==4)  TP=[1,0,0,1,1]; end;
if (m==5)  TP=[1,0,0,1,0,1]; end;
if (m==6)  TP=[1,0,0,0,0,1,1]; end;
if (m==7)  TP=[1,0,0,0,1,0,0,1]; end;
if (m==8)  TP=[1,0,0,0,1,1,1,0,1]; end;
if (m==12) TP=[1,0,0,0,0,0,1,0,1,0,0,1,1]; end;
if (m==16)
    TP=[1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,1,1]; end;
jj=1;
while (ii>0)&(ii<2.^m-1)&(jj<=ii)
     if (pwi<2.^(m-1)) pwi=pwi.*2;
        else
            pwi=pwi.*2;
            sh=conv2bin(pwi,m);
            nsh=length(TP)-length(sh);
            if (nsh>0)
                for ij=1:nsh sh=[0,sh]; end;
            end;
            pwibin=xor(sh',TP);
            pwi=convbin2dec(pwibin');
     end;
    jj=jj+1;
```

```
    end;
return;
```

Функція loggf2m(m, i) для обчислення логарифмів від значень елементів i, які належать полю Галуа *GF*(2^m)

```
%The logarifm function log2(i) for Galuas Field
GF(2^m),
%where m - the order of the field.
%Functions parameters are:
%1. m - order of the field. Possible value of
%this parameter are 3,4,5,6,7,8,12,16.
%2. i - logarifmic function argument.
%Possible vaues of this paraemter: 0<=i<=2^m-1
%Both m and i have to be integer values.
%This function is widely used in discrete mathematic
%for forming the codes of digital sequences.
%For example, Reed - Solomon code based on arithmetic
%operations in Galoas Fields.
%Examples of using this function:
%> loggf2m(8,23)
%ans =
%   129
%>> loggf2m(4,15)
%ans =
%    12
%See also:  powgf2m, prodgf2m, divgf2m,
%           sumgf2m, conv2bin, convbin2dec
function jj=loggf2m(m,ii)
    fd1='Wrong input papameter m, order o';
    fd2='f Galoas Field GF(2^m). This va';
    fd3='lue have to be integer value bot';
```

```
fd4='h from 3 to 8 or 12 or 16.';
df=[fd1,fd2,fd3,fd4];
lpp=m-floor(m);
if (lpp~=0) error (df); end;
if ((m<3)|((m>8)&(m~=12)&(m~=16))) error (df); end;
sd=(2.^m-1)-ii;
rt1='Wrong input papameter i, the log fu';
rt2='nction argument. It should be smaller, t';
rt3='hen maximum element of Galoas Field 2^m-';
rt4='1, where m - order of the Field, first fu';
rt5='nction parameter.';
rt=[rt1,rt2,rt3,rt4,rt5];
if (sd<0) error (rt); end;
if (ii==2.^m-1) pwi=0; end;
if (ii==0) pwi=1; end;
if (m==3)  TP=[1,0,1,1]; end;
if (m==4)  TP=[1,0,0,1,1]; end;
if (m==5)  TP=[1,0,0,1,0,1]; end;
if (m==6)  TP=[1,0,0,0,0,1,1]; end;
if (m==7)  TP=[1,0,0,0,1,0,0,1]; end;
if (m==8)  TP=[1,0,0,0,1,1,1,0,1]; end;
if (m==12) TP=[1,0,0,0,0,0,1,0,1,0,0,1,1]; end;
if (m==16) TP=[1,0,0,0,1,0,0,0,0,0,0,0,0,1,0,1,1]; end;
jj=0; cc=1;
if (ii==0) jj=2.^m-1; end;
if (ii==1) jj=0; end;
while (ii>1)&(ii<=(2.^m-1))&(cc~=ii)
    if (cc<2.^(m-1)) cc=cc.*2;
        else
            cc=cc.*2;
```

```
                sh=conv2bin(cc,m);
                nsh=length(TP)-length(sh);
                if (nsh>0)
                    for ij=1:nsh sh=[0,sh]; end;
                end;
                pwibin=xor(sh',TP);
                cc=convbin2dec(pwibin');
          end;
      jj=jj+1;
      end;
return;
```

Функція prodgf2m(m, i, j) для множення елементів i та j, які належать полю Галуа *GF*(2^m)

```
%The function for multiplication of elements
%of Galuas Field GF(2^m), ii and jj,
%where m - the order of the field.
%Functions parameters are:
%1. m - order of the field. Possible value of
%this parameter are 3,4,5,6,7,8,12,16.
%2. ii - first multiplier, it may be vector,
%jj - second multiplier.
%Possible vaues of this paraemter: 0<=i<=2^m-1
%Both m and jj have to be integer values,
%ii may be the vector of integer values.
%This function is widely used in discrete mathematic
%for forming the codes of digital sequences.
%For example, Reed - Solomon code based on arithmetic
%operations in Galoas Fields.
%Examples of using this function:
```

```
%>> prodgf2m(3,2,3)
%ans =
%     6
%>> prodgf2m(5,2,3)
%ans =
%     6
%>>
%>> prodgf2m(3,[2,5,6,3],5)
%ans =
%     1     7     3     4
%See also:  powgf2m,loggf2m, divgf2m, sumgf2m
function kk=prodgf2m(m,ii,jj)
    fd1='Wrong input papameter m, order o';
    fd2='f Galoas Field GF(2^m). This va';
    fd3='lue have to be integer value bot';
    fd4='h from 3 to 8, 12 or 16.';
    df=[fd1,fd2,fd3,fd4];
    lpp=m-floor(m);
    if (lpp~=0) error (df); end;
    if ((m<3)|((m>8)&(m~=12)&(m~=16))) error (df); end;
    rt1='Wrong input papameter i or j, the production fu';
    rt2='nction argument. They should be smaller, t';
    rt3='hen maximum element of Galoas Field 2^m-';
    rt4='1, where m - order of the Field, first fu';
    rt5='nction parameter.';
    rt=[rt1,rt2,rt3,rt4,rt5];
    nii=length(ii);
    for jjj=1:nii
        sd1(jjj)=(2.^m-1)-ii(jjj);
        if (sd1(jjj)<0) error (rt); end;
```

```
    end;
    sd2=(2.^m-1)-jj;
    if (sd2>(2.^m-1)) error (rt); end;
    cf=loggf2m(m,jj);
    for iii=1:nii
        if((jj==0)|(ii(iii)==0)) kk(iii)=0; else
            ss(iii)=loggf2m(m,ii(iii))+cf;
            if (ss(iii)>=(2.^m-1))
                    ss(iii)=ss(iii)-(2.^m-1); end;
            kk(iii)=powgf2m(m,ss(iii));
        end;
    end;
return;
```

Функція divgf2m(m, i, j) для ділення елементів i та j, які належать полю Галуа *GF*(2^m)

```
%The function for division of elements
%of Galuas Field GF(2^m), ii and jj,
%where m - the order of the field.
%Functions parameters are:
%1. m - order of the field. Possible value of
%this parameter are 3,4,5,6,7,8,12,16.
%2. ii - dividend, jj - divisor.
%Possible vaues of this paraemter: 0<=i<=2^m-1
%Both m and ii, jj have to be integer values.
%This function is widely used in discrete mathematic
%for forming the codes of digital sequences.
%For example, Reed - Solomon code based on arithmetic
%operations in Galoas Fields.
%Examples of using this function:
```

```
%>> divgf2m(5,2,3)
%ans =
%    29
%>> divgf2m(5,2,4)
%ans =
%    18
%>>
%>> divgf2m(4,[12,13,1,3,4],5)
%ans =
%    13     6    11    14    10
%See also:  powgf2m,loggf2m, prodgf2m, sumgf2m
function kk=divgf2m(m,ii,jj)
    fd1='Wrong input papameter m, order o';
    fd2='f Galoas Field GF(2^m). This va';
    fd3='lue have to be integer value bot';
    fd4='h from 3 to 8, 12 or 16.';
    df=[fd1,fd2,fd3,fd4];
    lpp=m-floor(m);
    if (lpp~=0) error (df); end;
    if ((m<3)|((m>8)&(m~=16)&(m~=16))) error (df); end;
    sd1=(2.^m-1)-ii; sd2=(2.^m-1)-jj;
    rt1='Wrong input papameter i or j, the deviding fu';
    rt2='nction argument. They should be smaller, t';
    rt3='hen maximum element of Galoas Field 2^m-';
    rt4='1, where m - order of the Field, first fu';
    rt5='nction parameter.';
    rt=[rt1,rt2,rt3,rt4,rt5];
    nii=length(ii);
    for jjj=1:nii
```

```
        sd1(jjj)=(2.^m-1)-ii(jjj);
        if (sd1(jjj)<0) error (rt); end;
    end;
    sd2=(2.^m-1)-jj;
    if (sd2>(2.^m-1)) error (rt); end;
    qq='Error! Division by zero. Check input data.';
    if (jj==0) error (qq); end;
    cf=loggf2m(m,jj);
    for iii=1:nii
        ss(iii)=loggf2m(m,ii(iii))-cf;
        if (ss(iii)<0) ss(iii)=ss(iii)+(2.^m-1); end;
        if (ii(iii)==0) kk(iii)=0; else
            kk(iii)=powgf2m(m,ss(iii)); end;
    end;
return;
```

Функція для множення поліномів, елементи яких належать полю Галуа GF(2^m)

```
%Function prodpolgf2m(m,vin1,vin2) for multiplying
%of two polynoms, vin1 and vin2, in the Galuas Field
%GF(2^m). m - order of the field. Possible value of
%this parameter are 3,4,5,6,7,8,12,16.
%Examples of using this function:
%>> prodpolgf2m(3,[4,5,1],[1,0,1])
%ans =
%     4     5     5     5     1
%>> prodpolgf2m(8,[49,148,249],[1,6,8])
%ans =
%    49    50    51   248   155
```

```
%See also:  powgf2m, loggf2m, prodgf2m,
%           divgf2m, sumgf2m
function vout=prodpolgf2m(m,vin1,vin2)
    n1=length(vin1); n2=length(vin2);
    fd1='Wrong input papameter m, order o';
    fd2='f Galoas Field GF(2^m). This va';
    fd3='lue have to be integer value bot';
    fd4='h from 3 to 8, 12 or 16.';
    df=[fd1,fd2,fd3,fd4];
    lpp=m-floor(m);
    if (lpp~=0) error (df); end;
    if ((m<3)|((m>8)&(m~=16)&(m~=16))) error (df); end;
    rt1='Wrong input papameter vin1 or vin2, th';
    rt2='e arguments of function for polynomial m';
    rt3='ultiplication. Its should be smaller, t';
    rt4='hen maximum element of Galoas Field 2^m-';
    rt5='1, where m - order of the Field, first fun';
    rt6='ction parameter.';
    rt=[rt1,rt2,rt3,rt4,rt5,rt6];
    for ij=1:n1
        sd1=(2.^m-1)-vin1(ij);
        if (sd1<0) error (rt); end;
    end;
    for ij=1:n2
        sd2=(2.^m-1)-vin2(ij);
        if (sd2<0) error (rt); end;
    end;
    if (n1>=n2)
       P1=vin1; P2=vin2;
```

```
        np1=n1;  np2=n2;
         else
           P1=vin2; P2=vin1;
           np1=n2;  np2=n1;
    end; %if (n1>=n2)
    RESPERV=zeros(1,np1);
    for jj=np2:-1:1
        cc=prodgf2m(m,P1,P2(jj));
        ZN=zeros(1,np2-jj);
        RES=[cc,ZN];
        if (jj<np2) RESPERV=[0,RESPERV]; end;
        RES=sumgf2m(m,RES,RESPERV);
        RESPERV=RES;
    end; %for (ii=1:np1)
    vout=RES;
return;
```

Функція для ділення поліномів, елементи яких належать полю Галуа *GF*(2^m)

```
%Function divpolgf2m(m,vin1,vin2) for deviding
%of two polynoms, vin1 and vin2, in the Galuas Field
%GF(2^m). m - order of the field. Possible value of
%this parameter are 3,4,5,6,7,8,12,16.
%vin1 - dividend, vin2 - divisor. THe length of vector
%vin1 have to be grater, than the length of vector vin2
%Examples of using this function:
%>> divpolgf2m(4,[12,13,6,14,15,7],[1,12,4,15])
%ans =
%     0    12     2     8
```

```
%     0     4     4     6
%>>
%>> divpolgf2m(4,[12,13,6,14,15,7],[1,4,15])
%ans =
%    12     8     8     9
%     0     0    12     9
%>>
function MOUT=divpolgf2m(m,vin1,vin2)
    n1=length(vin1); n2=length(vin2);
    fd1='Wrong input papameter m, order o';
    fd2='f Galoas Field GF(2^m). This va';
    fd3='lue have to be integer value bot';
    fd4='h from 3 to 8, 12 or 16.';
    df=[fd1,fd2,fd3,fd4];
    lpp=m-floor(m);
    if (lpp~=0) error (df); end;
    if ((m<3)|((m>8)&(m~=12)&(m~=16))) error (df); end;
    rt1='Wrong input papameter vin1 or vin2, th';
    rt2='e arguments of function for polynomial d';
    rt3='evision. Its should be smaller, t';
    rt4='hen maximum element of Galoas Field 2^m-';
    rt5='1, where m - order of the Field, first fun';
    rt6='ction parameter.';
    rt=[rt1,rt2,rt3,rt4,rt5,rt6];
    for ij=1:n1
        sd1=(2.^m-1)-vin1(ij);
        if (sd1<0) error (rt); end;
    end;
    for ij=1:n2
        sd2=(2.^m-1)-vin2(ij);
```

```
        if (sd2<0) error (rt); end;
    end;
    ty1='Wrong input vectors vin1 and vin2. Length of d';
    ty2='ividend, vector vin1, have to be grater, than t';
    ty3='he length of divisor, vector vin2.';
    ty=[ty1,ty2,ty3];
    if (n1<n2) error (ty); end;
    if (vin2(1)~=1) vin2=divgf2m(m,vin2,vin2(1)); end;
    DVD=vin1; nd=1000; ii=1; QOT=[]; dz=0; ndvd=n1;
ndvs=n2;
    while ((nd>=0)&(DVD(1)>=0))
        ndvdper=length(DVD);
        while (DVD(1)==0)&(ndvd>ndvs)
           DVD=DVD(2:ndvd);
           ndvd=length(DVD);
        end; %while (DVD(1)==0)&(DVD(ndvd)~=0)
        DVS=prodgf2m(m,vin2,DVD(1));
        ndvs=length(DVS); ndvd=length(DVD);
        dz=ndvdper-ndvd;
        if (dz>1)
                ZRES=zeros(1,dz-1);
                QOT=[QOT,ZRES,DVD(1)];
            else
                if (ndvd>=ndvs) QOT=[QOT,DVD(1)]; end;
        end;%if (dz>1)
        nd=length(DVD)-length(DVS);
        nzrs=ndvd-ndvs; DVS=[DVS,zeros(1,nzrs)];
        if (nd>=0)&(DVD(1)~=0) DVD=sumgf2m(m,DVD,DVS);
                else
                    ql=length(QOT);
```

```
                    if (DVD(1)==0) QOT=QOT(1:ql-1); end;
                    break;
                end;
    end %while (dn>=0)
    ndvd=length(DVD); nqot=length(QOT);
    DVD=DVD(2:ndvd);  dltn=ndvd-nqot-1;
    if (dltn>0) ZD=zeros(1,dltn); QOT=[ZD,QOT]; end;
    if (dltn<0) ZD=zeros(1,abs(dltn)); DVD=[ZD,DVD]; end;
    MOUT=[QOT;DVD];
return
```

Програма для обчислення значень поліноміальних функцій, елементи яких належать полю Галуа GF(2^m)

```
%Function calcpolgf2m(m,vin,k) for calculation the
%value of polynomial functions in the Galuas Field
%GF(2^m). m - order of the field, k - value of
%argument, vin - input vector, which described the
%polynomial function. For example, popolynom
%3*x^2+2*x+1 described by vector [3,2,1].
%Examples of using this function:
%>> calcpolgf2m(8,[19,93,1],27)
%ans =
%   140
%>> calcpolgf2m(8,[19,93,1],131)
%ans =
%     0
%See also:  powgf2m, loggf2m, prodgf2m,
%           divgf2m, sumgf2m, prodpolgf2m,
%           divpolgf2m
function sum=calcpolgf2m(m,vin,k)
    n=length(vin);
    fd1='Wrong input papameter m, order o';
```

```
    fd2='f Galoas Field GF(2^m). This va';
    fd3='lue have to be integer value bot';
    fd4='h from 3 to 8, 12 or 16.';
    df=[fd1,fd2,fd3,fd4];
    lpp=m-floor(m);
    if (lpp~=0) error (df); end;
    if ((m<3)|((m>8)&(m~=12)&(m~=16))) error (df); end;
    rt1='Wrong input parameters of vector vin, th';
    rt2='e function for polynomial val';
    rt3='ue calculation. All coefficients of polyno';
    rt4='m should be smaller, then maximum element o';
    rt5='f Galoas Field 2^m-1, where m - order of t';
    rt6='he field, first function parameter.';
    rt=[rt1,rt2,rt3,rt4,rt5,rt6];
    for ij=1:n
        sd=(2.^m-1)-vin(ij);
        if (sd<0) error (rt); end;
    end;
    qt1='Wrong input papameter k, th';
    qt2='e arguments of function for polynomial val';
    qt3='ue calculation. Its should be smaller, t';
    qt4='hen maximum element of Galoas Field 2^m-';
    qt5='1, where m - order of the Field, first fun';
    qt6='ction parameter.';
    qt=[qt1,qt2,qt3,qt4,qt5,qt6];
    sd2=(2.^m-1)-k;
    if (sd2<0) error (qt); end;
    sum=vin(n); c=1;
    for ii=n-1:-1:1
        c=prodgf2m(m,c,k);
        sum=sumgf2m(m,sum,prodgf2m(m,c,vin(ii)));
    end;
return
```

Результати тестування наведених програмних модулей, призначених для виконання алгебраїчних та поліноміальних операцій у полях Галуа GF(2^m)

```
>> sumgf2m(8,40,22)
ans =
    62
>> sumgf2m(4,10,12)
ans =
     6
>> sumgf2m(4,[10,11,12,13],[9,8,3,11])
ans =
     3     3    15     6
>> sumgf2m(8,[100,110,120,130],[90,80,30,110])
ans =
    62    62   102   236
>> sumgf2m(4,[12,13,6,14,15,7],[1,4,15])
??? Error using ==> sumgf2m
Wrong input vectors ii and jj. The vectors must have same
length.
>> powgf2m(4,10)
ans =
     7
>> powgf2m(5,12)
ans =
    14
>> powgf2m(8,251)
ans =
   216
>> powgf2m(4,20)
```

```
??? Error using ==> powgf2m
Wrong input papameter i, the power function argument. It
should be smaller, then maximum element of Galoas Field
2^m-1, where m - order of the Field, first function
parameter.
>> loggf2m(4,7)
ans =
    10
>> loggf2m(5,14)
ans =
    12
>> loggf2m(8,216)
ans =
   251
>> prodgf2m(3,2,6)
ans =
     7
>> prodgf2m(3,11,13)
??? Error using ==> prodgf2m
Wrong input papameter i or j, the production function
argument. They should be smaller, then maximum element of
Galoas Field 2^m-1, where m - order of the Field, first
function parameter.
>> prodgf2m(4,11,13)
ans =
     6
>> prodgf2m(8,6,49)
ans =
   166
>> prodgf2m(8,8,49)
```

```
ans =
   149
>> prodgf2m(4,[3,5,8,12],13)
ans =
     4    12     2     3
>> prodgf2m(3,[3,5,2],3)
ans =
     5     4     6
>> prodgf2m(3,[3,0,2],3)
ans =
     5     0     6
>>
>> divgf2m(8,161,25)
ans =
    51
>> divgf2m(8,166,49)
ans =
     6
>> divgf2m(4,8,11)
ans =
    14
>> divgf2m(4,3,13)
ans =
    12
>> divgf2m(4,[12,13,1,3,4],5)
ans =
    13     6    11    14    10
>> divgf2m(8,[120,130,100,134,240],5)
ans =
   24    42    77   140    48
>> divgf2m(4,[12,0,1,0,4],5)
ans =
```

```
    13     0    11     0    10
>> divgf2m(4,[12,13,1,3,4],0)
??? Error using ==> divgf2m
Error! Division by zero. Check input data.
>> prodpolgf2m(4,[4,5,1,13,12,10],[1,2,0,5,1])
ans =
    4   13   11    8    3   1   10   4    8   10
>> prodpolgf2m(8,[49,148,249],[1,6,8])
ans =
    49    50    51   248   155
>> prodpolgf2m(8,[154,134,125],[111,122])
ans =
   166   221    70   155
>> prodpolgf2m(8,[154,134,201,136,212,57,33],[111,122])
ans =
   166   221   161   107   120   201   139   129
>> prodpolgf2m(8,[154,134,201,136,212,57,33],[111,122,151,144])
ans =
   166  221  135  165  237  53  21  207  230  103
>> divpolgf2m(4,[12,13,6,14,15,7],[1,12,4,15])
ans =
     12    2     8
     4     4     6
>> pr=prodpolgf2m(4,[12,2,8],[1,12,4,15])
pr =
    12    13    6    10    11    1
>> res=sumgf2m(4,pr,[4,4,7])
??? Error using ==> sumgf2m
Wrong input vectors ii and jj. The vectors must have same
length.
>> res=sumgf2m(4,pr,[0,0,0,4,4,7])
```

```
res =
    12    13     6    14    15     6
>> divpolgf2m(4,pr,[1,12,4,15])
ans =
     12     2     8
     0     0     0
>> divpolgf2m(4,res,[1,12,4,15])
ans =
     12     2     8
     4     4     7
>> powgf2m(8,-6)
ans =
    54
>> powgf2m(8,-2)
ans =
    71
>> powgf2m(3,-5)
ans =
     4
>> powgf2m(4,-12)
ans =
     8
>> powgf2m(5,-20)
ans =
     7
>> calcpolgf2m(8,[203,224,236,229,240,5,61,81,228],2)
ans =
   246
>> calcpolgf2m(8,[203,224,236,229,240,5,61,81,228],4)
ans =
```

```
   207
>> calcpolgf2m(8,[203,224,236,229,240,5,61,81,228],8)
ans =
   255
>> calcpolgf2m(8,[203,224,236,229,240,5,61,81,228],16)
ans =
   213

>> calcpolgf2m(8,[19,93,1],16)
ans =
    82
>> calcpolgf2m(8,[19,93,1],1)
ans =
    79
>> calcpolgf2m(8,[19,93,1],142)
ans =
   108
>> calcpolgf2m(8,[19,93,1],71)
ans =
    37
>> calcpolgf2m(8,[19,93,1],173)
ans =
    52
>> calcpolgf2m(8,[19,93,1],216)
ans =
    85
>> calcpolgf2m(8,[19,93,1],216)
ans =
    85
>> calcpolgf2m(8,[19,93,1],54)
```

```
ans =
     0
>> calcpolgf2m(8,[19,93,1],27)
ans =
   140
>> calcpolgf2m(8,[19,93,1],131)
ans =
     0
>>
>> c=calcpolgf2m(8,[181,246],powgf2m(8,-6))
c =
   211
>> d=calcpolgf2m(8,[181,246],powgf2m(8,-8))
d =
   184
>> divgf2m(8,c,93)
ans =
     6
>> divgf2m(8,d,93)
ans =
    30
```

ДОДАТОК О. Програма для формування систематичних кодів Ріда – Соломона та декодування їхніх кодових послідовностей

```
%Fuction for calculation the sequences of RS Codes
% and for decoding of these codes. Input parameters:
%vin - sequence of volues, which have to be coded or
%decoded in Galuas Fild GF(2^m).
%Only value from 0 to (2^m)-1 for vector vin is allows.
%Possible value of m are: 3,4,5,6,7,8,12 and 16.
%Another function parameters are:
%ne - amount of detected errors. Possible value of this
%parameter is 1,2,3,4,5,6.
%nop - number of necessary operation.
%Possible values of this parameter:
%1 - forming of RS code.
%2 - decoding RS code sequence.
%In this case output vector is decoded sequence.
%Error positions in decoded sequence are also noted.
%Examples of using this function
%>> RSC([12,14,7,10],4,2,1)
%ans =
%  12  14   7  10   8  13   6   4
%>> RSC([46,53,39,27],6,3,1)
%ans =
%  46  53  39  27  21  48  38  24  36  53
%>> RSC([14,6,13,7],5,3,1)
%ans =
%  14   6  13   7  29  29  27  21   9  16
%>> c=RSC([14,6,13,7],5,3,1)
%c =
%  14   6   13   7   29   29   27   21   9    16
%>> RSC([14,6,13,7],5,3,1)
%ans =
%  14   6   13   7   29   29   27   21   9    16
```

```
%>> RSC(c,5,3,2)
%RS-code sequence is correct. Coded sequence is:
%14   6  13   7
%ans =
%    14     6    13     7
%>> c(2)=5; c(4)=10;
%>> RSC(c,5,3,2)
%RS-code is incorrect, multiple error in the
%7-th and 9-th digits have been detedted.
%Correct code sequence is:
%14   6  13   7  29  29  27  21   9  16
%Coded sequence is:
%14   6  13   7
%ans =
%    14     6    13     7
%>> c(3)=15;
%RS-code is incorrect, multiple error in the
%7-th, 8-th and 9-th digits have been detedted.
%Correct code sequence is:
%14  26  31  17  29  29  27  21   9  16
%Coded sequence is:
%14  26  31  17
%ans =
%    14    26    31    17
%>> RSC(c,5,3,2)
%RS-code sequence is incorrect and can not be
%corrected, perhaps more than 3 errors in it.
%Please check input data and try again.
%ans =
%     []
%See also:  powgf2m, loggf2m, prodgf2m,
%           divgf2m, sumgf2m, prodpolgf2m,
%           divpolgf2m
function vout=RSC(vin,m,ne,nop)
```

```
nv=length(vin);
df1='Wrong value of Galuas field order m. Possible v';
df2='alues are: 3,4,5,6,7,8, 12 and 16.
 df3='Check input data!';
dfstr=[df1,df2,df3];
if ((m~=3)&(m~=4)&(m~=5)&(m~=6)&(m~=7)&...
    (m~=7)&(m~=8)&(m~=12)&(m~=16)) error (dfstr); end;
dd1='Wrong input vector! Elements of vector have t';
dd2='o be in range from 0 to (2^m)-1. ';
ddstr=[dd1,dd2,df3];
for ii=1:nv
    if(((vin(ii)<0)|(vin(ii)>(2.^m)-1)))
                              error (ddstr); end;
end; %for ii=1:nv
dg1='Wrong value of amount of errors ne. Possible v';
dg2='alues are: 1,2,3,4,5 or 6. Check input data!';
dgstr=[dg1,dg2];
if ((ne~=1)&(ne~=2)&(ne~=3)&(ne~=4)&(ne~=5)&(ne~=6))
    error (dgstr); end;
db1='Wrong value of type of operation nop. Possible v';
db2='alues are 1 or 2. Check input data!';
dbstr=[dg1,dg2];
if ((nop~=1)&(nop~=2)) error (dbstr); end;
dw1='Wrong length of input vector vin. Possible le';
dw2='ngth for coding sequence is from 3 to 16 and f';
dw3='or decoding sequence from 5 to 28.';
dwstr=[dw1,dw2,dw3];
if ((nop==1)&((nv<3)|(nv>16))) error (dwstr); end;
if ((nop==2)&((nv<5)|(nv>28))) error (dwstr); end;
wst1='Wrong parameter ne, amount of erros in code. I';
wst2='t should be smaller, than the amount of digits.';
wst=[wst1,wst2];
if ((nop==2)&(ne>nv)) error (wst); end;
ww1='Wrong number of detected errors ne f';
```

```
    ww2='or defined Galuas Field GF(2^m). ';
    ww3='Number of errors, multiplied by 2, shoul';
    ww4='d to be smaller, than the maximum elemen';
    ww5='t of  Field (2^m-1). Check input data!'
   www=[ww1,ww2,ww3,ww4,ww5];
   ter=2.*ne; maxel=2^m-1;
   if ((nop==2)&(ter>maxel)) error (www); end;
   ew1='Wrong length of vector of RS-code. It s';
   ew2='hould be smaller, than the maximum element o';
   ew3='f Galuas Field GF(2^m). Check input data!';
   ew=[ew1,ew2,ew3];
   if ((nop==2)&(nv>maxel)) error (ew); end;
   if ((nop==1)&((nv+2*ne)>maxel)) error (ew); end;
   if (nop==1)
       TPRS=[1]; kk=2.*ne; pwr=1;
       for ii=1:kk
           TPRS=prodpolgf2m(m,TPRS,[1,powgf2m(m,ii)]);
       end;%for ii=1:kk
       ntp=length(TPRS); NZ=zeros(1,ntp-1);
       VDIV=[vin,NZ];
       RES_DIV=divpolgf2m(m,VDIV,TPRS);
       REST=RES_DIV(2,:);
       lr=length(REST); cc=1;
       while (cc<=lr)&(REST(cc)==0)
           REST(cc)=[]; lr=length(REST);
       end;
       lv=length(VDIV);
       for tt=0:lr-1
           VDIV(lv-tt)=REST(lr-tt);
       end;
       vout=VDIV;
   end;%if (nop==1)
   if (nop==2)
%Defining of sindroms errors coefficients
```

```
c(1,1)=2; c(1,2)=powgf2m(m,2);
if(ne>1)
    c(1,3)=powgf2m(m,3);
    c(1,4)=powgf2m(m,4);
end;%if(ne>1)
if(ne>2)
    c(1,5)=powgf2m(m,5);
    c(1,6)=powgf2m(m,6);
end;%if(ne>2)
if(ne>3)
    c(1,7)=powgf2m(m,7);
    c(1,8)=powgf2m(m,8);
end;%if(ne>3)
if(ne>4)
    c(1,9)=powgf2m(m,9);
    c(1,10)=powgf2m(m,10);
end;%if(ne>4)
if(ne==6)
    c(1,11)=powgf2m(m,11);
    c(1,12)=powgf2m(m,12);
end;%if(ne==6)
for ii=2:nv-1
    for jj=1:2.*ne
        c(ii,jj)=prodgf2m(m,c(ii-1,jj),c(1,jj));
    end;%for jj=2:2.*ne
end;%for ii=2:nv-1
for jj=1:2.*ne
    S(jj)=0;
    for ii=1:nv
        if(ii<nv) mn=c(nv-ii,jj); else mn=1; end;
        S(jj)=sumgf2m(m,prodgf2m(m,...
                        vin(ii),mn),S(jj));
    end;%for ii=1:nv
end;%for jj=1:2.*ne
```

```
        ee=0; iii=1;
        while (ee==0)&(iii<=2.*ne)
            if(S(iii)~=0) ee=1; end;
            iii=iii+1;
        end;%while (ee==0)&(iii<=2.*ne)
        if (ee==0)
            vout=vin(1:nv-2*ne);
            Vidsp=num2str(vout);
            disp('RS-code sequence is correct. C...
                                        oded sequence is:');
            disp(Vidsp);
        end;
%Correction of sequences with single error
        if ((ne==1)&(ee==1))
            u=loggf2m(m,S(2))-loggf2m(m,S(1));
            if (u<0) u=u+(2^m-1); end;
            if (u<nv)
                niu=divgf2m(m,prodgf2m(m,S(1),...
                      S(1)),S(2));
                corr=vin; corr(nv-u)=sumgf2m(m,...
                      corr(nv-u),niu);
                nd=u+1; NDdisp=num2str(nd);
                disp('RS-code is incorrect, 1 e...
                           rror detedted in the');
                rt='-th digit. ';
                rtr=[NDdisp, rt]; disp(rtr);
                disp('Correct code sequence is:');
                Vidsp=num2str(corr); disp(Vidsp);
                disp('Coded sequence is:');
                vout=corr(1:nv-2*ne);
                Vidsp2=num2str(vout); disp(Vidsp2);
            end; %if (u<nv)
            if (u>=nv)
                vout=[];
```

```
            disp('RS-code sequence is incor...
                                rect and can not be');
            disp('corrected, perhaps more th...
                                an 1 error in it.');
            disp('Please check input d...
                                ata and try again.');
        end; %if (u<nv)
    end; %if ((ne==1)&(ee==1))
    if ((ne>1)&(ee==1))
    %Correction of sequences with multiple errors
    %1. Berlekamp - Messay algorithm;
        LBD0=1; B=[1,0]; ri=0; L=0; mi=-1;
        LBD_POL=zeros(1,2.*ne+1);
        LBD_POL(2.*ne+1)=LBD0;
        while (ri<2*ne)
            DELT=0;
            for id=0:L
                if (id==0)
                    DELT_COMP=S(ri-id+1);
                    DELT=DELT_COMP;
                        else
                            DELT_COMP=prodgf2m(m,...
                            LBD_POL(NLBD_POL-id),...
                            S(ri-id+1));
                            DELT=sumgf2m(m,DELT,...
                            DELT_COMP);
                end;%if (id==0)
            end; %for id=0:L
            if (DELT~=0)
                POLB=prodgf2m(m,B,DELT);
                NPOLB=length(POLB);
                NLBD_POL=length(LBD_POL);
                DN=NLBD_POL-NPOLB;
                if(DN>=0)
```

```
                    ZD=zeros(1,DN);
                    BPLUS=[ZD,POLB];
                end;%if(DN>0)
                LBD_POL_n=sumgf2m(m,LBD_POL,BPLUS);
                if (L<(ri-mi))
                    L_n=ri-mi; mi=ri-L; L=L_n;
                    DELT_INV=divgf2m(m,1,DELT);
                    B=prodgf2m(m,LBD_POL,DELT_INV);
                end;%if (L<(ri-mi))
                LBD_POL=LBD_POL_n;
            end; %if (DELT~=0)
            LB=length(B);
            B=[B(2:LB),0]; ri=ri+1;
        end; %while (ri<2*ne)
        nui=1; Chng_Ord_LBD=0;
        while(LBD_POL(nui)==0)
            LBD_POL(nui)=[];
            Chng_Ord_LBD=1;
        end; %while(LBD_POL(nui)==0)
        OrdLBD=length(LBD_POL)-1; nne=OrdLBD;
%2. Special analyzing of situation, when 1 error is
%detected in the RS-code sequencw for correction of
%multiply errors;
        if (nne==1)
            u=loggf2m(m,S(2))-loggf2m(m,S(1));
            if (u<0) u=u+(2^m-1); end;
            niu=divgf2m(m,prodgf2m(m,S(1),S(1)),S(2));
            corr=vin;
            corr(nv-u)=sumgf2m(m,corr(nv-u),niu);
            nd=u+1; NDdisp=num2str(nd);
            disp('RS-code is incorrect,...
                  1 error detedted in the');
            rt='-th digit. ';
            rtr=[NDdisp, rt]; disp(rtr);
```

```
                disp('Correct code sequence is:');
                Vidsp=num2str(corr); disp(Vidsp);
                disp('Coded sequence is:');
                vout=corr(1:nv-2*ne);
                Vidsp2=num2str(vout); disp(Vidsp2);
            end;%if (nne==1)
%3. Detecting of unrecognized errors.
             if (OrdLBD~=L)&(Chng_Ord_LBD==0)
                NDdsp=num2str(ne);
                disp('RS-code sequence is incor...
                                      rect and can not be');
                qwq1=('corrected, perhaps more than ');
                qwq2=(' errors in it.');
                qwq=[qwq1, NDdsp, qwq2]; disp(qwq);
                disp('Please check input data a...
                                      nd try again.');
                vout=[];
             end; %if (OrdLBD~=L)&(Chng_Ord_LBD==0)
%Detecting of recognized errors and correction of code
sequence.
            if (OrdLBD==L)
%4. Finding the roots of errors locator polynom.
                ii=0; jj=1;
                if (nne>1)
                  while (ii<=(nv-1))&(jj<=nne)
                     cc=calcpolgf2m(m,LBD_POL,...
                               powgf2m(m,(-1)*ii));
                     if (cc==0)
                         u(jj)=ii;
                         jj=jj+1;
                      end; %if (cc==0)
                      ii=ii+1;
                  end; %while (ii<=(nv-1))&(jj<=nne)
              end; %if (nne>1)
```

```
            end;%if (OrdLBD==L)
            if ((jj==1)&(nne>1))
                Nsp=num2str(ne);
                disp('RS-code sequence is incor...
                                rect and can not be');
                qtq1=('corrected, perhaps more than ');
                qtq2=(' errors in it.');
                qtq=[qtq1, Nsp, qtq2];
                disp(qtq);
                disp('Please check input data a...
                                    nd try again.');
                vout=[];
            end;%if ((jj==1)&(nne>1))
            if ((jj>1)&(nne>1))
%5. Finding the polynom of errors values.
                LS=length(S); SP=S(LS:-1:1);
                EVP=prodpolgf2m(m,SP,LBD_POL);
                EVPL=length(EVP);
                EVP=EVP(2*ne-1:EVPL);
                nni=1;
                while(EVP(nni)==0) EVP(nni)=[]; end;
%6. Finding the derivation of errors locator polynom.
                 if (OrdLBD==2) LBD_DER=LBD_POL(2); end;
                 if ((OrdLBD==3)|(OrdLBD==4))
                     LBD_DER=[LBD_POL
                       (OrdLBD-2),0,LBD_POL(OrdLBD)];
                end;% if ((OrdLBD==3)|(OrdLBD==4))
                 if ((OrdLBD==5)|(OrdLBD==6))
                     LBD_DER=[LBD_POL(OrdLBD-...
                           4),0,LBD_POL(OrdLBD-2),...
                                0,LBD_POL(OrdLBD)]; ]
                end;% if ((OrdLBD==5)|(OrdLBD==6))
%7. Finding the value of errors.
                LNTHU=length(u);
```

```
                for ik=1:LNTHU
                    ck=powgf2m(m,(-1)*u(ik));
                    c1=calcpolgf2m(m,EVP,ck);
                    c2=calcpolgf2m(m,LBD_DER,ck);
                    EVal(ik)=divgf2m(m,c1,c2);
                end; %for ik=1:LNTHU
%7. Forming  the vector of errors.
                Err_Vect=zeros(1,nv);
                for is1=1:LNTHU
                      Err_Vect(nv-u(is1))=EVal(is1);
               end; %for is1=1:LNTHU
%8. Correction of wrong code sequence.
                corr=sumgf2m(m,vin,Err_Vect);
                vout=corr(1:nv-2*ne);
%9. Forming of output message.
                disp('RS-code is incorrect, multi...
                          ple error in the');
                rt1='-th'; rt2=' digits have be...
                          en detedted.';
                rt3=' and '; rt4=', ';
                rtr=[];
                for ik=1:LNTHU
                    nd=u(ik)+1; NDdisp=num2str(nd);
                    if (ik<LNTHU-1)
                          rtr=[rtr, NDdisp, rt1, rt4];
                    end; %if (ik<LNTHU-1)
                    if (ik==LNTHU-1)
                          rtr=[rtr, NDdisp, rt1];
                    end;% if (ik==LNTHU-1)
                    if (ik==LNTHU)
                          rtr=[rtr, rt3, NDdisp, rt1];
                    end; %if (ik==LNTHU)
                end;% for ik=1:LNTHU
                rtr3=[rtr, rt2];
```

```
                disp(rtr3);
                disp('Correct code sequence is:');
                Vd=num2str(corr); disp(Vd);
                disp('Coded sequence is:');
                vout=corr(1:nv-2*ne);
                Vidsp2=num2str(vout); disp(Vidsp2);
            end;%if ((jj>1)&(nne>1))
        end; %if ((ne>1)&(ee==1))
    end;%if (nop==2)
return;
```

Результати тестування програми

```
>> C=[213, 224, 234, 229, 240];
>> D=RSC(C,8,2,1)
D =
   213   224   234   229   240     5    61    81   228
>> F=RSC(D,8,2,2)
RS-code sequence is correct. Coded sequence is:
213  224  234  229  240
F =
   213   224   234   229   240
>> D(1)=203;
>> F=RSC(D,8,2,2)
RS-code is incorrect, 1 error detedted in the
9-th digit.
Correct code sequence is:
213  224  234  229  240    5   61   81  228
Coded sequence is:
213  224  234  229  240
F =
   213   224   234   229   240
>> D(3)=236;
```

```
>> F=RSC(D,8,2,2)
RS-code is incorrect, multiple error in the
7-th and 9-th digits have been detedted.
Correct code sequence is:
213  224  234  229  240    5   61   81  228
Coded sequence is:
213  224  234  229  240
F =
   213   224   234   229   240
>>
>> D(5)=242;
>> F=RSC(D,8,2,2)
RS-code sequence is incorrect and can not be
corrected, perhaps more than 2 errors in it.
Please check input data and try again.
F =
     []
>> C=[46, 53, 39, 27];
>> D=RSC(C,6,3,1)
D =
   46   53   39   27   21   48   38   24   36   53
>> F=RSC(D,6,3,2)
RS-code sequence is correct. Coded sequence is:
46  53  39  27
F =
    46    53    39    27
>> D(1)=48;
>> F=RSC(D,6,3,2)
RS-code is incorrect, 1 error detedted in the
10-th digit.
Correct code sequence is:
46  53  39  27  21  48  38  24  36  53
Coded sequence is:
```

```
46  53  39  27
F =
    46    53    39    27
>> D(2)=55;
>> F=RSC(D,6,3,2)
RS-code is incorrect, multiple error in the
9-th and 10-th digits have been detedted.
Correct code sequence is:
46  53  39  27  21  48  38  24  36  53
Coded sequence is:
46  53  39  27
F =
    46    53    39    27
>> D(3)=35;
>> F=RSC(D,6,3,2)
RS-code is incorrect, multiple error in the
8-th, 9-th and 10-th digits have been detedted.
Correct code sequence is:
46  53  39  27  21  48  38  24  36  53
Coded sequence is:
46  53  39  27
F =
    46    53    39    27
>> F=RSC(D,6,3,2)
RS-code sequence is incorrect and can not be
corrected, perhaps more than 3 errors in it.
Please check input data and try again.
F =
     []
>> C=[13,15,12,11,13];
>> D=RSC(C,4,5,1);
>> D
D =
```

```
  13  15  12  11  13  11  3  11  14  14  4  11  6  0  15
>> F=RSC(D,4,5,2);
RS-code sequence is correct. Coded sequence is:
13  15  12  11  13
>> D(1)=10; D(2)=14; D(5)=8;
>> F=RSC(D,4,5,2);
RS-code is incorrect, multiple error in the
11-th, 14-th and 15-th digits have been detedted.
Correct code sequence is:
13  15  12  11  13  11   3  11  14  14   4  11   6   0  15
Coded sequence is:
13  15  12  11  13
>> D(3)=15; D(4)=14;
>> F=RSC(D,4,5,2);
RS-code is incorrect, multiple error in the
11-th, 12-th, 13-th, 14-th and 15-th digits have been
detedted.
Correct code sequence is:
13  15  12  11  13  11   3  11  14  14   4  11   6   0  15
Coded sequence is:
13  15  12  11  13
>> C=[46, 53, 39, 27, 34];
>> D=RSC(C,6,6,1)
D =
  Columns 1 through 13
    46  53  39  27  34  14  62  15  40  23  33  21   0
  Columns 14 through 17
    52  55   4  46
>> D(1)=48; D(2)=55; D(3)=35;
>> F=RSC(D,6,6,2)
RS-code is incorrect, multiple error in the
15-th, 16-th and 17-th digits have been detedted.
Correct code sequence is:
```

```
46 53 39 27 34 14 62 15 40 23 33 21  0  52  55  4  46
Coded sequence is:
46  53  39  27  34
>> D(4)=48; D(5)=55; D(6)=35;
>> F=RSC(D,6,6,2)
RS-code is incorrect, multiple error in the
12-th, 13-th, 14-th, 15-th, 16-th and 17-th digits have
been detedted.
Correct code sequence is:
46 53 39 27 34 14 62 15 40 23 33 21 0  52  55  4  46
Coded sequence is:
46  53  39  27  34
F =
    46    53    39    27    34
>> C=[18,15,12,11,13];
>> D=RSC(C,4,3,1)
??? Error using ==> rsc
Wrong input vector! Elements of vector have to be in
range from 0 to (2^m)-1. Check input data!
>> D=RSC(C,5,3,1)
D =
       18  15  12  11  13   8   25  15   2   22    9
>> C=[3,5,2,1,4];
>> D=RSC(C,3,3,1)
??? Error using ==> rsc
Wrong length of vector of RS-code. It should be
smaller, than the maximum element of Galuas Field
GF(2^m). Check input data!
>> C=[13,15,12,11,14,17,9,10,7,4,12,3,5,6,7,8,4,5,3];
>> D=RSC(C,4,3,2)
??? Error using ==> rsc
Wrong input vector! Elements of vector have to be in
range from 0 to (2^m)-1. Check input data!
```

```
>> C=[13,15,12,11,14,7,9,10,7,4,12,3,5,6,7,8,4,5,3];
>> D=RSC(C,4,3,2)
??? Error using ==> rsc
Wrong length of vector of RS-code. It should be
smaller, than the maximum element of Galuas Field
GF(2^m). Check input data!
>> C=[13,15,12,11,3,9,7];
>> D=RSC(C,4,5,1)
??? Error using ==> rsc
Wrong length of vector of RS-code. It should be
smaller, than the maximum element of Galuas Field
GF(2^m). Check input data!
>> D=RSC(C,4,4,1)
D =
  Columns 1 through 13
    13  15  12  11 3  9  7  12  4  5  2  4  13  6  4
  Columns 14 through 15
>> D(3)=10;
>> F=RSC(D,4,1,2)
RS-code is incorrect, 1 error detedted in the
13-th digit.
Correct code sequence is:
13 15 12 11  3  9  7  12  4  5  2  4  13  6  4
Coded sequence is:
13 15 12 11  3  9  7  12  4  5  2  4  13
F =
   13  15  12  11   3   9   7  12   4   5   2   4  13
>> D(4)=5;
>> F=RSC(D,4,4,2)
RS-code is incorrect, 1 error detedted in the
13-th digit.
Correct code sequence is:
```

```
13 15 12 11 3  9  7  12  4  5  2  4  13  6  4
Coded sequence is:
13  15  12  11   3   9   7
F =
    13    15    12    11     3     9     7
>> F=RSC(D,4,2,2)
RS-code is incorrect, multiple error in the
12-th and 13-th digits have been detedted.
Correct code sequence is:
13  15  12  11  3  9  7  12  4  5  2  4  13  6   4
Coded sequence is:
13  15  12  11   3   9   7  12   4   5   2
F =
    13  15  12  11   3   9   7  12   4   5   2
>> F=RSC(D,4,4,2)
RS-code is incorrect, multiple error in the
12-th and 13-th digits have been detedted.
Correct code sequence is:
13 15 12  11  3  9  7  12  4  5  2  4  13  6   4
Coded sequence is:
13  15  12  11   3   9   7
F =
    13    15    12    11     3     9     7
>> D(1)=11; D(2)=13;
>> F=RSC(D,4,4,2)
RS-code is incorrect, multiple error in the
12-th, 13-th, 14-th and 15-th digits have been
detedted.
Correct code sequence is:
13  15  12  11   3   9   7  12  4  5  2  4  13  6   4
Coded sequence is:
13  15  12  11   3   9   7
```

```
F =
    13   15   12   11    3    9    7
>> D(7)=5;
>> F=RSC(D,4,4,2)
RS-code sequence is incorrect and can not be
corrected, perhaps more than 4 errors in it.
Please check input data and try again.
F =
   []
>>
```

Результати оцінки часу виконання програми RSC для різної кількості помилок, які виправляються, для різної довжини кодових комбінацій та для різних порядків полів Галуа

```
function TESTRSC
    C=[13,15,12,11,13];
    t1=cputime;
    D=RSC(C,4,3,1);
    t2=cputime;
    dt1=t2-t1
    D(1)=10;
    F=RSC(D,4,3,2);
    t3=cputime;
    dt2=t3-t2
    D(2)=14; D(5)=8;
    F=RSC(D,4,3,2);
    t4=cputime;
    dt3=t4-t3
    C=[213, 224, 234, 229, 240];
```

```
D=RSC(C,8,2,1);
t5=cputime;
dt4=t5-t4
D(1)=203;
F=RSC(D,8,2,2);
t6=cputime;
dt5=t6-t5
D(3)=236;
F=RSC(D,8,2,2);
t7=cputime;
dt6=t7-t6
C=[46, 53, 39, 27, 34];
D=RSC(C,6,6,1);
t8=cputime;
dt7=t8-t7
D(1)=48; D(2)=55; D(3)=35;
F=RSC(D,6,6,2);
t9=cputime;
dt8=t9-t8
D(4)=48; D(5)=55; D(6)=35;
F=RSC(D,6,6,2);
t10=cputime;
dt9=t10-t9
C=[23341, 31433, 18165];
D=RSC(C,16,1,1);
t11=cputime;
dt10=t11-t10
F=RSC(D,16,1,2);
t12=cputime;
dt11=t12-t11
D(2)=31438;
F=RSC(D,16,1,2);
t13=cputime;
```

```
    dt12=t13-t12
    C=[2341, 3433, 1865];
    D=RSC(C,12,2,1);
    t14=cputime;
    dt13=t14-t13
    D(1)=3438;
    F=RSC(D,12,2,2);
    t15=cputime;
    dt14=t15-t14
return

>> TESTRSC
dt1 =
    0.3200
RS-code is incorrect, 1 error detedted in the
11-th digit.
Correct code sequence is:
13  15  12  11  13  15   6   0   8   1   0
Coded sequence is:
13  15  12  11  13
dt2 =
    0.9020
RS-code is incorrect, multiple error in the
7-th, 10-th and 11-th digits have been detedted.
Correct code sequence is:
13  15  12  11  13  15   6   0   8   1   0
Coded sequence is:
13  15  12  11  13
dt3 =
    1.8320
dt4 =
    5.6990
RS-code is incorrect, 1 error detedted in the
```

```
9-th digit.
Correct code sequence is:
213  224  234  229  240    5   61   81  228
Coded sequence is:
213  224  234  229  240
dt5 =
   10.7850
RS-code is incorrect, multiple error in the
7-th and 9-th digits have been detedted.
Correct code sequence is:
213  224  234  229  240    5   61   81  228
Coded sequence is:
213  224  234  229  240
dt6 =
   32.0960
dt7 =
    6.4390
RS-code is incorrect, multiple error in the
15-th, 16-th and 17-th digits have been detedted.
Correct code sequence is:
46 53 39 27 34 14 62 15 40 23 33 21  0  52  55  4  46
Coded sequence is:
46  53  39  27  34
dt8 =
   28.3510
RS-code is incorrect, multiple error in the
12-th, 13-th, 14-th, 15-th, 16-th and 17-th digits have
been detedted.
Correct code sequence is:
46 53 39 27 34 14 62 15 40 23 33 21  0 52  55  4  46
Coded sequence is:
46  53  39  27  34
dt9 =
```

```
   41.3300
dt10 =
  815.7130
RS-code sequence is correct. Coded sequence is:
23341  31433  18165
dt11 =
  665.8470
RS-code is incorrect, 1 error detedted in the
4-th digit.
Correct code sequence is:
23341  31433  18165  12867  27643
Coded sequence is:
23341  31433  18165
dt12 =
  1.2733e+003
dt13 =
   90.9710
RS-code is incorrect, 1 error detedted in the
7-th digit.
Correct code sequence is:
2341  3433  1865  2620  2035  63   629
Coded sequence is:
2341  3433  1865
dt14 =
  183.6940
>>
```

Програма для оцінки імовірності приймання спотвореної комбінації коду Ріда – Соломона із помилками, які не можливо виправити

```
function RSERROR(m,k,t)
    pb=1e-5:1e-5:0.2;
    n=k+2.*t;
```

```
    d1='Wrong parameters, code c';
    d2= an not be created. Check input data!';
    dd=[d1, d2];
    if (n>((2.^m)-1)) error (dd); end;
    S=0;
    for TET=t+1:n
        CNTET=fact(n)./(fact(TET).*fact(n-TET));
        Sn1=(1-(1-pb).^m).^TET;
        Sn2=((1-pb).^m).^(n-TET);
        S=S+CNTET.*Sn1.*Sn2;
    end;
    plot(pb,S);
    grid on; hold on;
return;

function nf=fact(nn);
nf=1;
for ii=1:nn
    nf=nf.*ii;
end;
return
```

ДОДАТОК П. Програма для формування згорткових кодів з різними параметрами та декодування їхніх кодових послідовностей

```
%Fuction for calculation the sequences of
%convolutional
%codes and for decoding its. Input parameters:
%vin - sequence of values, which have to be %coded or
decoded.
%Only values 0 or 1 for the vector vin are %allows.
%notsk - the nubmer of task. The possible values %of
this
%parameter are follows:
%1 - task of coding;
%2 - task of decoding.
%kp - code parameters. The possible values of
%this parameter are follows:
%1 - constraint length K=3,
%redundancy ratio 1/n = 1/2;
%2 - constraint length K=4,
%redundancy ratio 1/n = 1/3;
%3 - constraint length K=5,
%redundancy ratio 1/n = 1/3;
%Command line to calling this function:
%convcode(vin,kp,notssk)
%Limits to the length of vector vin sequences:
%20 - for coding;
%60 - for decoding.
% Some examples of using this function:
% >> c=[1,0,0,1,0,1,1,0];
% >> s=cnvcode(c,1,1)
% s =
```

```
%   Columns 1 through 10
%    1   1   1   1   0   1   0   0   1   0
%   Columns 11 through 16
%     1     0     1     1     0     0
% >> f=cnvcode(s,1,2)
% f =
%    1    0    0    1    0    1    1    0
% See also: parity
function vout=convcode(vin,kp,notsk)
    nv=length(vin);
    df1='Wrong value of task parameter notsk. ';
    df2='It should be equil 1 or 2. Ple...
        ase check input data!';
    dfstr=[df1,df2];
    if ((notsk~=1)&(notsk~=2)) error (dfstr); end;
    dk1='Wrong input vector! For code...
        d sequence number of vector el';
    dk2='ements shoud be smaller tha...
        n 10. Please check input data!';
    dkstr=[dk1,dk2];
    if ((notsk==1)&(nv>20)) error (dkstr); end;
    dd1='Wrong input vector! For deco...
        ded sequence number of vector el';
    dd2='ements shoud be smaller tha...
        n 10. Please check input data!';
    ddstr=[dd1,dd2];
    if ((notsk==2)&(nv>60)) error (ddstr); end;
    dw1='Wrong input vector! All elemen...
        ts of this vectior should be eq';
    dw2='uil 0 or 1. Please check input data!';
```

```
ddwtr=[dw1,dw2];
for ii=1:nv
    if((vin(ii)<0)|(vin(ii)>1)) error (ddwtr); end;
end; %for ii=1:nv
for iii=1:nv
    vin_inv(iii)=vin(nv-iii+1);
end; %for iii=1:nv
if (notsk==1)
    Vout=[];
    if (kp==1)
        BUF=[0, 0, 0];
        for iform=1:nv
            To_Buf=vin_inv(iform);
            Buf2=[To_Buf,BUF]; BUF=Buf2;
            BUF(4)=[];
            S2=xor(BUF(1),BUF(3));
            S1=parity(BUF);
            V=[Vout, S1, S2]; Vout=V;
        end; %for iform=1:nv
    end;%if (kp==1)
    if (kp==2)
        BUF=[0, 0, 0, 0];
        for iform=1:nv
            To_Buf=vin_inv(iform);
            Buf2=[To_Buf,BUF]; BUF=Buf2;
            BUF(5)=[];
            S1=parity(BUF);
            S2=parity(BUF(1:3));
            S3=parity(BUF(2:4));
            V=[Vout, S1, S2, S3]; Vout=V;
```

```
        end; %for iform=1:nv
    end;%if (kp==2)
    if (kp==3)
        BUF=[0, 0, 0, 0, 0];
        for iform=1:nv
            To_Buf=vin_inv(iform);
            Buf2=[To_Buf,BUF]; BUF=Buf2;
            BUF(6)=[];
            S1=parity(BUF);
            S2=parity(BUF(1:4));
            S3=parity(BUF(2:5));
            V=[Vout, S1, S2, S3]; Vout=V;
        end; %for iform=1:nv
    end;%if (kp==3)
end;%if (notsk==1)
if (notsk==2)
    if (kp==1)
        Mcorr1=[0,-1,0,-1; 1,-1,1,-1; ...
                  -1,0,-1,0; -1,1,-1,1];
        Mcorr2=[0,-1,1,-1; 1,-1,0,-1; ...
                 -1,1,-1,0; -1,0,-1,1];
        Mcorr3=[0,-1,1,-1; 1,-1,0,-1; ...
                 -1,0,-1,1; -1,1,-1,0];
        Mcorr=Mcorr1; Mcorr(:,:,2)=Mcorr2;
        Mcorr(:,:,3)=Mcorr3;
        dr1='Wrong input vector for deco...
             ding! The amount of elem';
        dr2='ents of this vector shou...
             ld be even. Please check in...
             put data!';
```

```
   drwtr=[dr1,dr2];
   if (mod(nv,2)~=0)
            error (drwtr); end;
   NST=4; lout=nv/2;
     Nsimb=2; SUMMET=-2;
end%if (kp==1)
if (kp==2)
   Mcorr1=[0,-1,-1,-1,0,-1,-1,-1;...
              1,-1,-1,-1,1,-1,-1,-1;...
          -1,0,-1,-1,-1,0,-1,-1;
              -1,1,-1,-1,-1,1,-1,-1;...
           -1,-1,0,-1,-1,-1,0,-1;...
              -1,-1,1,-1,-1,-1,1,-1;...
            -1,-1,-1,0,-1,-1,-1,0;...
              -1,-1,-1,1,-1,-1,-1,1];
   Mcorr2=[0,-1,-1,-1,1,-1,-1,-1;...
              1,-1,-1,-1,0,-1,-1,-1;...
            -1,1,-1,-1,-1,0,-1,-1;...
              -1,0,-1,-1,-1,1,-1,-1;...
            -1,-1,1,-1,-1,-1,0,-1;...
            -1,-1,0,-1,-1,-1,1,-1;...
         -1,-1,-1,0,-1,-1,-1,1;...
              -1,-1,-1,1,-1,-1,-1,0];
   Mcorr3=[0,-1,-1,-1,0,-1,-1,-1;
              1,-1,-1,-1,1,-1,-1,-1;...
           -1,1,-1,-1,-1,1,-1,-1; ...
             -1,0,-1,-1,-1,0,-1,-1;...
           -1,-1,1,-1,-1,-1,1,-1; ...
             -1,-1,0,-1,-1,-1,0,-1;...
          -1,-1,-1,0,-1,-1,-1,0;...
```

```
            -1,-1,-1,1,-1,-1,-1,1];
   Mcorr4=[0,-1,-1,-1,1,-1,-1,-1;...
             0,-1,-1,-1,1,-1,-1,-1;...
          -1,1,-1,-1,-1,0,-1,-1;...
             -1,1,-1,-1,-1,0,-1,-1;...
          -1,-1,1,-1,-1,-1,0,-1;...
             -1,-1,1,-1,-1,-1,0,-1;...
           -1,-1,-1,0,-1,-1,-1,1;...
             -1,-1,-1,0,-1,-1,-1,1];
   Mcorr=Mcorr1; Mcorr(:,:,2)=Mcorr2;
   Mcorr(:,:,3)=Mcorr3;
     Mcorr(:,:,4)=Mcorr4;
   dr1='Wrong input vector for deco...
        ding! The amount of elem';
   dr2='ents of this vector shou...
        ld be devided by 3. Ple';
   dr3='ase check input data!';
   drwtr=[dr1,dr2,dr3];
   if (mod(nv,3)~=0) ...
        error (drwtr); end;
   NST=8; lout=nv/3;
     Nsimb=3; SUMMET=-3;
end%if (kp==2)
if (kp==3)
   Mcorr1=[0,-1,-1,-1,-1,-1,-1,-1,...
             0,-1,-1,-1,-1,-1,-1,-1;...
           1,-1,-1,-1,-1,-1,-1,-1,...
             1,-1,-1,-1,-1,-1,-1,-1;...
          -1,0,-1,-1,-1,-1,-1,-1,...
             -1,0,-1,-1,-1,-1,-1,-1;...
```

```
        -1,1,-1,-1,-1,-1,-1,-1,...
           -1,1,-1,-1,-1,-1,-1,-1;...
        -1,-1,0,-1,-1,-1,-1,-1,...
           -1,-1,0,-1,-1,-1,-1,-1;...
        -1,-1,1,-1,-1,-1,-1,-1,...
           -1,-1,1,-1,-1,-1,-1,-1;...
        -1,-1,-1,0,-1,-1,-1,-1,...
           -1,-1,-1,0,-1,-1,-1,-1;...
        -1,-1,-1,1,-1,-1,-1,-1,...
           -1,-1,-1,1,-1,-1,-1,-1;...
        -1,-1,-1,-1,0,-1,-1,-1,...
           -1,-1,-1,-1,0,-1,-1,-1;...
        -1,-1,-1,-1,1,-1,-1,-1,...
           -1,-1,-1,-1,1,-1,-1,-1;...
        -1,-1,-1,-1,-1,0,-1,-1,...
           -1,-1,-1,-1,-1,0,-1,-1;...
        -1,-1,-1,-1,-1,1,-1,-1,...
           -1,-1,-1,-1,-1,1,-1,-1;...
        -1,-1,-1,-1,-1,-1,0,-1,...
           -1,-1,-1,-1,-1,-1,0,-1;...
        -1,-1,-1,-1,-1,-1,1,-1,...
           -1,-1,-1,-1,-1,-1,1,-1;...
        -1,-1,-1,-1,-1,-1,-1,0,...
           -1,-1,-1,-1,-1,-1,-1,0;...
        -1,-1,-1,-1,-1,-1,-1,1,...
           -1,-1,-1,-1,-1,-1,-1,1];
Mcorr2=[0,-1,-1,-1,-1,-1,-1,-1,...
          1,-1,-1,-1,-1,-1,-1,-1;...
        1,-1,-1,-1,-1,-1,-1,-1,...
          0,-1,-1,-1,-1,-1,-1,-1;...
```

```
 -1,1,-1,-1,-1,-1,-1,-1,...
    -1,0,-1,-1,-1,-1,-1,-1;...
 -1,0,-1,-1,-1,-1,-1,-1,...
    -1,1,-1,-1,-1,-1,-1,-1;...
 -1,-1,1,-1,-1,-1,-1,-1,...
     -1,-1,0,-1,-1,-1,-1,-1;...
  -1,-1,0,-1,-1,-1,-1,-1,...
     -1,-1,1,-1,-1,-1,-1,-1;...
 -1,-1,-1,0,-1,-1,-1,-1,...
     -1,-1,-1,1,-1,-1,-1,-1;...
-1,-1,-1,1,-1,-1,-1,-1,...
  -1,-1,-1,0,-1,-1,-1,-1;...
-1,-1,-1,-1,1,-1,-1,-1,...
  -1,-1,-1,-1,0,-1,-1,-1;...
-1,-1,-1,-1,0,-1,-1,-1,...
  -1,-1,-1,-1,1,-1,-1,-1;...
-1,-1,-1,-1,-1,0,-1,-1,...
  -1,-1,-1,-1,-1,1,-1,-1;...
-1,-1,-1,-1,-1,1,-1,-1,...
  -1,-1,-1,-1,-1,0,-1,-1;...
-1,-1,-1,-1,-1,-1,0,-1,...
  -1,-1,-1,-1,-1,-1,1,-1;...
-1,-1,-1,-1,-1,-1,1,-1,...
   -1,-1,-1,-1,-1,-1,0,-1;...
-1,-1,-1,-1,-1,-1,-1,1,...
   -1,-1,-1,-1,-1,-1,-1,0;...
-1,-1,-1,-1,-1,-1,-1,1,...
   -1,-1,-1,-1,-1,-1,-1,1];
Mcorr3=[0,-1,-1,-1,-1,-1,-1,-1,...
          0,-1,-1,-1,-1,-1,-1,-1;...
```

```
1,-1,-1,-1,-1,-1,-1,-1,...
   1,-1,-1,-1,-1,-1,-1,-1;...
-1,1,-1,-1,-1,-1,-1,-1,...
  -1,1,-1,-1,-1,-1,-1,-1;...
-1,0,-1,-1,-1,-1,-1,-1,...
   -1,0,-1,-1,-1,-1,-1,-1;...
-1,-1,1,-1,-1,-1,-1,-1,...
   -1,-1,0,-1,-1,-1,-1,-1;...
-1,-1,0,-1,-1,-1,-1,-1,...
   -1,-1,0,-1,-1,-1,-1,-1;...
-1,-1,-1,0,-1,-1,-1,-1,...
   -1,-1,-1,0,-1,-1,-1,-1;...
-1,-1,-1,1,-1,-1,-1,-1,...
   -1,-1,-1,1,-1,-1,-1,-1;...
-1,-1,-1,-1,1,-1,-1,-1,...
   -1,-1,-1,-1,1,-1,-1,-1;...
-1,-1,-1,-1,0,-1,-1,-1,...
   -1,-1,-1,-1,0,-1,-1,-1;...
-1,-1,-1,-1,-1,0,-1,-1,...
   -1,-1,-1,-1,-1,0,-1,-1;...
-1,-1,-1,-1,-1,1,-1,-1,...
   -1,-1,-1,-1,-1,1,-1,-1;...
-1,-1,-1,-1,-1,-1,0,-1,...
   -1,-1,-1,-1,-1,-1,0,-1;...
-1,-1,-1,-1,-1,-1,1,-1,...
   -1,-1,-1,-1,-1,-1,1,-1;...
-1,-1,-1,-1,-1,-1,-1,1,...
   -1,-1,-1,-1,-1,-1,-1,1;...
-1,-1,-1,-1,-1,-1,-1,1,...
   -1,-1,-1,-1,-1,-1,-1,0];
```

```
Mcorr4=[0,-1,-1,-1,-1,-1,-1,-1,...
          1,-1,-1,-1,-1,-1,-1,-1;...
        0,-1,-1,-1,-1,-1,-1,-1,...
          1,-1,-1,-1,-1,-1,-1,-1;...
       -1,1,-1,-1,-1,-1,-1,-1,...
          -1,0,-1,-1,-1,-1,-1,-1;...
      -1,1,-1,-1,-1,-1,-1,-1,...
         -1,0,-1,-1,-1,-1,-1,-1;...
      -1,-1,1,-1,-1,-1,-1,-1,...
         -1,-1,0,-1,-1,-1,-1,-1;...
      -1,-1,1,-1,-1,-1,-1,-1,...
         -1,-1,0,-1,-1,-1,-1,-1;...
      -1,-1,-1,0,-1,-1,-1,-1,...
         -1,-1,-1,1,-1,-1,-1,-1;...
      -1,-1,-1,0,-1,-1,-1,-1,...
         -1,-1,-1,1,-1,-1,-1,-1;...
      -1,-1,-1,-1,1,-1,-1,-1,...
         -1,-1,-1,-1,0,-1,-1,-1;...
      -1,-1,-1,-1,1,-1,-1,-1,...
         -1,-1,-1,-1,0,-1,-1,-1;...
      -1,-1,-1,-1,-1,0,-1,-1,...
         -1,-1,-1,-1,-1,1,-1,-1;...
      -1,-1,-1,-1,-1,-1,0,-1,...
        -1,-1,-1,-1,-1,-1,1,-1;...
      -1,-1,-1,-1,-1,-1,0,-1,...
         -1,-1,-1,-1,-1,-1,1,-1;...
      -1,-1,-1,-1,-1,-1,-1,1,...
         -1,-1,-1,-1,-1,-1,-1,1;...
```

```
            -1,-1,-1,-1,-1,-1,-1,0,...
               -1,-1,-1,-1,-1,-1,-1,0];
    Mcorr=Mcorr1; Mcorr(:,:,2)=Mcorr2;
    Mcorr(:,:,3)=Mcorr3;
      Mcorr(:,:,4)=Mcorr4;
    dr1='Wrong input vector for deco...
              ding! The amount of elem';
    dr2='ents of this vector shou...
              ld be devided by 3. Ple';
    dr3='ase check input data!';
    drwtr=[dr1,dr2,dr3];
    if (mod(nv,3)~=0)...
              error (drwtr); end;
    NST=16; lout=nv/3;
     Nsimb=3; SUMMET=-3;
end%if (kp==3)
STNew=1; Vout=[]; Err=0;
for ifd=1:lout
     ST(ifd)=STNew;
     csec=vin_inv(Nsimb*(ifd-...
              1)+1:Nsimb*ifd);
     CompMET(ifd)=100;
        for nstr=1:NST
              SIGN=Mcorr(nstr,ST(ifd),...
                     2:Nsimb+1);
              SIGN_CMP=reshape(SIGN,1,...
                     Nsimb,1);
            MET=sum(SIGN);
            if (MET~=SUMMET)
                  CompM(nstr)=sum(xor...
```

```
                    (SIGN_CMP,csec));
                if(CompMET(ifd)>...
                    CompM(nstr))
                     CompMET(ifd)=...
                          CompM(nstr);
                   STNew=nstr;
               end;%if(CompMET(ifd)>
           end;%if (MET~=SUMMET)
       end; %for nstr=1:lout
       MET_RES=sum(CompMET);
       Vout=[Vout,Mcorr(STNew,ST(ifd),1)];
end; %for ifd=1:lout
Pos_Chng=lout; ST_Chng=ST(lout);
Vout2(1:lout)=0; Numb_Err=0;
while(MET_RES>3)&(sum(xor(Vout,...
                          Vout2))>0)
       Cur_Pos_Chng=Pos_Chng;
         Numb_Err=1;
       for ifd_n=lout-3:-1:1
           if((CompMET(ifd_n)==0)&...
                 (CompMET(ifd_n+1)~=0)&...
                 (CompMET(ifd_n-1)==0))
              Pos_Chng=ifd_n+1;
                 ST_Chng=ST(ifd_n+1);
            end;%if(CompMET(ifd_n)==0)&...
       end; % for ifd_n=lout-3:-1:1
       if (Pos_Chng==Cur_Pos_Chng)
            Pos_Chng=Pos_Chng+1; end;
       ST_Chng=ST(Pos_Chng);
       for nstr=1:NST
```

```
            SIGN_N=Mcorr(nstr,...
                 ST(Pos_Chng),2:Nsimb+1);
            SIGN_CMP_N=reshape(SIGN_N,...
                 1,Nsimb,1);
            MET_N=sum(SIGN_CMP_N);
            if(MET_N~=SUMMET)&...
                      (MET_N~=ST_Chng)...
                 CompMET(ifd_n)=...
                      CompM(nstr); end;
       Mout=Vout;
          Vout2=Vout(1:Pos_Chng);
 end;%for nstr=1:NST
for ifd_chng=Pos_Chng+1:lout
    ST(ifd_chng)=STNew;
    csec=vin_inv(Nsimb*...
              (ifd_chng-1)+...
        1:Nsimb*ifd_chng);
    CompMET(ifd_chng)=100;
    for nstr=1:NST
         SIGN=Mcorr(nstr,ST(ifd_chng),
              2:Nsimb+1);
         SIGN_CMP=reshape(SIGN,1,...
              Nsimb,1);
       MET=sum(SIGN);
       if (MET~=SUMMET)
              CompM(nstr)=...
                 sum(xor(SIGN_CMP,csec));
              if(CompMET(ifd_chng)>...
                           CompM(nstr))
                 CompMET(ifd_chng)=...
```

```
                                    CompM(nstr);
                                STNew=nstr;
                            end;
                                %if(CompMET(ifd)>CompM(nstr))
                        end;%if (MET~=SUMMET)
                    end; %for nstr=1:lout
                      Vout2=[Vout2,Mcorr(STNew,...
                                        ST(ifd_chng),1)];
                end; %for ifd_chng=Pos_Chng+1:lout
                MET_RES_NEW=sum(CompMET);
                if(MET_RES_NEW<MET_RES)
                    MET_RES=MET_RES_NEW;
                    Vout=Vout2;
                end;% if(MET_RES_NEW<MET_RES)
            end;% while((MET_RES>5)&(CompM(lout)>1)&...
        end;% if (notsk==2)
        nvout=length(Vout);
        out_err1='Coded sequence of conventi';
        out_err2='onal code have some errors!';
        out_err=[out_err1,out_err2];
        if ((notsk==2)&(Numb_Err==1)) disp (out_err); end;
        for ii2=1:nvout vout(ii2)=Vout(nvout-ii2+1); end;
%for ii2=1:nvout
return;
```

Програма parity, призначена для обчислення суми чисел двійкової послідовності за модулем 2

```
%Function for calcilation the parity of binary vector.
%All elements of input vectror should to be equil 0 or 1.
```

```
% Some examples of using this function:
% >> c1=[1,0,1,1,0];
% >> c2=[1,1,0,1,1];
% >> c3=[1,0,0,1,1];
% >> s=parity(c1)
% s =
%     1
% >> s=parity(c2)
% s =
%     0
% >> s=parity(c3)
% s =
%     1
% >> s=parity([2,1,0,1,0])
% Error using parity (line 22)
% Wrong input vector! All elements of this vectior
% should be
% equil 0 or 1. Please check input data!
function v_p=parity(vin)
    nv=length(vin);
    dw1='Wrong input vector! All elements of this...
             vectior should be eq';
    dw2='uil 0 or 1. Please check input data!';
    ddwtr=[dw1,dw2];
    for ii=1:nv
        if((vin(ii)<0)|(vin(ii)>1)) error (ddwtr); end;
    end; %for ii=1:nv
    n=0;
    for iii=1:nv
        if(vin(iii)==1)
```

```
            if (n==1) n=0; else n=1; end; %if(n==1)
        end; %if(vin(iii)==1)
    end; %for iii=1:nv
    v_p=n;
return
```

Результати тестування програми convcode

```
>> c=[1,0,0,1,0,1,1,0];
>> s=cnvcode(c,2,1)
s =
   Columns 1 through 10
    1   1   0   1   1   1   0   1   0   0
   Columns 11 through 20
    0   1   0   0   0   1   0   0   0   1
   Columns 21 through 24
     1     0     0     0
 f=cnvcode(s,2,2)
 f =
    1    0    0    1   0   1   1   0
 >> s=cnvcode(c,3,1)
 s =
   Columns 1 through 10
     1    0    0    0   1   0   1   0   1   0
   Columns 11 through 20
     1    1    0    0   0   1   0   0   0   1
   Columns 21 through 24
     1     0     0     0
 >> f=cnvcode(s,3,2)
 f =
    1    0    0    1   0   1   1   0
```

```
>> s(13:14)=1
s =
  Columns 1 through 10
    1    0    0    0    1    0    1    0    1    0
  Columns 11 through 20
    1    1    1    1    0    1    0    0    0    1
  Columns 21 through 24
    1     0     0     0
>> f=cnvcode(s,3,2)
f =
    1    0    0    1    0    1    1    0
>> s=cnvcode(c,3,1)
s =
  Columns 1 through 10
    1   0   0   0   1   0   1   0   1   0
  Columns 11 through 20
    1   1   0   0   0   1   0   0   1
  Columns 21 through 24
    1     0     0     0
>>  s(22:23)=1
  s =
  Columns 1 through 10
    1   0   0   0   1   0   1   0   1   0
  Columns 11 through 20
    1   1   0   0   0   1   0   0   0   1
  Columns 21 through 24
    1     1     1     0
>> f=cnvcode(s,3,2)
  f =
    1    0    0    1    0    1    1    0
```

```
>> c=[1,0,1,0,0,1,0,1,0,1,0,0,1,0,0,1,0,1,0,1];
>> s=cnvcode(c,3,1)
 s =
   Columns 1 through 10
     1     0     0     0     1     0     1     0     0     0
   Columns 11 through 20
     1     0     0     0     0     0     0     1     0     0
   Columns 21 through 30
     0     1     0     0     0     1     0     1     0     0
   Columns 31 through 40
     1     1     1     0     1     0     1     0     0     0
   Columns 41 through 50
     1     0     0     0     0     0     0     1     0     0
   Columns 51 through 60
     0     1     0     0     1     1     1     0     1     1
>> s(43:45)=1
 s =
   Columns 1 through 10
     1     0     0     0     1     0     1     0     0     0
   Columns 11 through 20
     1     0     0     0     0     0     0     1     0     0
   Columns 21 through 30
     0     1     0     0     0     1     0     1     0     0
   Columns 31 through 40
     1     1     1     0     1     0     1     0     0     0
   Columns 41 through 50
     1     0     1     1     1     0     0     1     0     0
   Columns 51 through 60
     0     1     0     0     1     1     1     0     1     1
>> f=cnvcode(s,3,2)
```

```
 Coded sequence of conventional code have
 some errors!
 f =
  Columns 1 through 10
      0     0    0    0    0    0    0    0    0     1
  Columns 11 through 20
      1     0    0    0    1    1    0    1    0     1
>> c=[1,0,0,2,0,1,1,0];
>> s=cnvcode(c,1,1)
 Error using cnvcode_vit (line 137)
 Wrong input vector! All elements of this
 vectior should be equil 0 or 1.
 Please check input data!
```

ДОДАТОК Р. Розрахунок поліноміальних коефіцієнтів для передавальної функції скінченного автомату, який відповідає моделі згорткового коду з параметрами K = 5, n = 1/3, з використанням функцій символьного процесора системи науково-технічних рохрахунків MatLab

```
>> syms D L N

>> XNN4=D^23*L^13*N^9-2*D^22*L^12*N^8-...

D^21*L^11*N^7*(L*N^2-1)+...

D^20*L^11*N^6*(N*(2*N-L+1)+2)-...

D^19*L^10*N^5*(2+N*(L*(L*N^3-1)-L+N+2))+...

D^18*L^10*N^5*((N^2*(1+2*L*N+L)-1)-6*N)+...

D^17*L^8*N^4*(1-L*N*(N*(1-2*L)+...

L*(N^2*(L*N^2-1)-1)))+...

2*D^17*L^9*N^4*(3*L^2*N-N+1)+...

D^16*L^8*N^3*(L*N*(3-N*(L*N^2-N-1))+...

2*(L*N*(1+N^2*(1-L))-2))+...

D^15*L^8*N^3*((2-N^2*(L+1))+L*N^2*(L*N+8))+...

D^14*L^7*N^2*((N^4*(1-2*L^2)+L*N^2*(1+...

N*(L+1))-1)-N*(L^2*N^2+6*L*N*(1-L*N^2)+2*N+4))+...

D^13*L^6*N^2*(L*(4*(1-L*N^2)-L*N^3*(3*L*N+4))-...

N^2*(L^2*N^2*(L*N-1)-L*(N*(N^2+1)-2)+3))-...

D^12*L^5*N^3*(L^3*N^3*(N+4)-2*(L*N*(L*(1-N))+1)+...

L*(3-L*N^2)+L^2*(L*N^2*(2-N)+2*(L*N+2)))+...

D^11*L^5*N^2*(L*N*(L*(N^2*(L*N-1)+N-2)+...

2*(1-L*N^2)-L^3*N^4-L^2*N^2*(N^3+N-1)-...

L*N*(N^2-1)+1))+...

D^10*L^5*N^2*(-L^2*N^3*(3+(N+1))+...
```

```
L*N^2*(N-3)+2*L*N+1)+...
D^9*L^5*N^2*(L*N^3*(L*N+1)-L*N*(N-5)+N*(N-1)-1)+...
D^8*L^4*N*(-L^2*N^3*(2*N+3)+L*N*(2*N^2+N+2)-N+1)+...
D^7*L^4*N^2*(2*L*N*(N-4)+3*L+1)-...
D^6*L^2*N*(L^2*N*(L^2*N-3)+1)+...
D^5*L^3*N*(L^2*N^3+(1-N*(N*(L+1)-...
1)))+D^4*L^3*N^2*(N*(L-1)+1)-...
D^3*L*(N-1)+D^2*L*N*(1-L*N)+1;
>> XND1=-D^19*L^12*N^8+D^18*L^11*N^7+...
4*D^17*L^11*N^8-...
D^16*L^10*N^6*(4*N+3)+D^15*L^9*N^5*(3-4*L*N^3)+...
D^14*L^8*N^4*(1+L*N^2*(3*N+4))+...
D^13*L^8*N^4*(L*N^4+(L+1)*N^2-5*N+2)+...
D^12*L^7*N^3*(2*(1+L*N^3)-3*N)+...
D^11*L^6*N^2*(L*N^4*(3*L-1)+2*(1-...
L*N^2))+D^10*L^7*N^4*(3-L*N^2)+...
D^9*L^5*N^3*(L^2*N^3-L*N^2+2)+...
D^8*L^5*N^4*(L*N+3)-D^7*L^4*N^2*(3*N+2)+...
D^6*L^3*N*(2-L*N^3)+D^4*L^2*N^2;
>> XNNOM_N=XNN4*XND1;
>> XNNOM_N2=expand(XNNOM_N);
>> XNNOM=collect(XNNOM_N2, D);
>> XND2=-D^19*L^12*N^8+2*D^18*L^11*N^7+...
4*D^17*L^11*N^8-...
D^16*L^10*N^6*(N^3+5*N+4)+...
D^15*L^9*N^5*(3+N-4*L*N^3)+...
```

```
D^14*L^8*N^4*(L*N^5+5*L*N^3+2*L*N^2+1)+...
D^13*L^8*N^4*(L*N^4+(L+1)*N^2-6*N-2)+...
D^12*L^7*N^3*(3*L*N^3-L*N^2-3*N+2)+...
D^11*L^6*N^2*(L*N^4*(3*L-1)+N^2*(L*(N-1)+1)+2)+...
D^10*L^7*N^4*(3-L*N^2)+...
D^9*L^6*N^2*(L*N^2-N+2)+...
D^8*L^5*N^4*(L*(N+1)+3)+D^7*L^4*N^2*(N*(L+3)+1)+...
D^6*L^3*N*(2-L*N^3)+D^4*L^2*N^2;
>> XND3=D^22*L^13*N^9-D^21*L^11*N^8-...
4*D^20*L^12*N^9+...
D^19*L^11*N^7*(3+N*(4-L))+...
D^18*L^10*N^6*(L*N*(1+4*N^2)-3)+...
D^17*L^9*N^5*(L*N^2*(N*(3-L)+2)-1)+...
D^16*L^9*N^5*(N*(L*(N*(N^2-1)-3)-N+5)+2)+...
D^15*L^9*N^5*(3-4*L*N^3)+...
D^14*L^7*N^3*(N*(L^2*N^2*(2*N+1)+...
L*(N*(N^2+2)+1))-2)+...
D^13*L^7*N^5*(L*N*(L*N^5*(N^2+1)+...
N*(N-7)-2)-1+L*N^2)+...
D^12*L^6*N^3*(L*N^3*(3-L*(N+1))+N*(N-5)+2)-...
D^11*L^6*N^2*(L*N^2*(N^2*(3*L+1)+1)+3*N-2)+...
D^10*L^5*N^3*(L*N*(L*(2-N^2)+1)+3*N+2)+...
D^9*L^4*N^2*(L*N*(L^2*N^2-((2*L-1)*N^2))-2)+...
D^8*L^5*N^4*(2*L*N+1)-D^7*L^3*N^2*(L*(N+1)+N)-...
D^6*L^3*N*(L*N^3-2)+D^4*L^2*N^2;
>> XNDEN_N=XND2*XND3;
```

```
>> XNDEN_N2=expand(XNDEN_N);
>> XNDEN=collect(XNDEN_N2, D);
>> XCNN=-D^11*L^7*N^3+D^9*L^6*N^4*(1-N^2)-...
D^8*L^5*N^3-D^7*L^5*N^3*(N-2)+D^6*L^4*N^2+...
D^5*L^4*N^3-D^4*L^2*N*(1+L*N)+D^3*L^2*N^2+...
D^2*L*(1-L*N);
>>  XCHN=2*D^11*L^6*N^3-2*D^10*L^5*N^2-...
2*D^8*L^5*N^2+2*D^7*L^4*N*(1-L*N^2)+ ...
D^6*L^4*N^2*(L*N+2)-D^5*L^4*N^2+...
D^4*L^3*N^2+D^3*L^2*N*(1-L*N)+D^2*L^2*N-D*L^2*N+L;
>> XCNHD=-D^10*L^7*N^5+D^9*L^6*N^4+...
D^8*L^6*N^5-D^7*L^5*N^3*(1+N)+...
D^6*L^4*N^2+D^5*L^3*N*(1-L*N^2)+...
D^4*L^4*N^3-D*L*N+1;
>> XCNOM_1=(1-D^3*L*N)*XNNOM*XCNN+XCHN*XNDEN;
>> XCDEN_1=(1-D^3*L*N)*XNDEN*XCNHD;
>> XCNOM_2=expand(XCNOM_1);
>> XCNOM=collect(XCNOM_2, D);
>> XCDEN_2=expand(XCDEN_1);
>> XCDEN=collect(XCDEN_2, D);
>> XBC=-D^9*L^6*N^3+D^7*L^5*N^5-D^3*L^3*N^2;
>> XBNNOM=-D^11*L^6*N^4+D^9*L^6*N^4*(1+N^2)-...
D^8*L^5*N^3-D^7*L^5*N^4+D^3*L^3*N^2-D^2*L^2*N;
>> XBHNOM=D^10*L^5*N^2-D^7*L^4*N-D^6*L^4*N^2+...
D^5*L^4*N^2-D^2*L*N;
>> XBNOM1=XBC*XCNOM*XNDEN*(1-D*L*N)*(1-D^3*L*N)+...
(1-D^3*L*N)*XCDEN*XBNNOM*XNNOM+...
(D*L*N-1)*XNDEN*XCDEN*XBHNOM;
```

```
>> XBDEN1=D^2*L*(1-D^3*L*N)*(1-D*L*N)*XNDEN*XCDEN;
>> XBNOM_2=expand(XBNOM1);
>> XBNOM=collect(XBNOM_2, D);
>> XBDEN_2=expand(XBDEN1);
>> XBDEN=collect(XBDEN_2, D);
>> XLNOM1=D^3*L^2*N^2*XCNOM*XNDEN+...
(D^5*L^3*N^2-D^3*L^2*N^2+D^2*L*N)*XCDEN*XNNOM;
>> XLDEN1=(1-D*L*N)*XCDEN*XNDEN;
>> XLNOM_2=expand(XLNOM1);
>> XLNOM=collect(XLNOM_2, D);
>> XLDEN_2=expand(XLDEN1);
>> XLDEN=collect(XLDEN_2, D);
>> XDNOM1=XLDEN-D^2*L*N*XLNOM;
>> XDDEN1=D^2*L*N*XLDEN;
>> XGNOM1=XLDEN+(D^4*L^2*N-D^2*L*N)*XLNOM;
>> XGDEN1=D^2*L*N*XLDEN;
>> XFNOM1=XLNOM*XNDEN-D^2*L*N*XNNOM*XLDEN;
>> XFDEN1=D^2*L*N*XLDEN*XNDEN;
>> XDNOM_2=expand(XDNOM1);
>> XDNOM=collect(XDNOM_2, D);
>> XDDEN_2=expand(XDDEN1);
>> XDDEN=collect(XDDEN_2, D);
>> XGNOM_2=expand(XGNOM1);
>> XGNOM=collect(XGNOM_2, D);
>> XGDEN_2=expand(XGDEN1);
>> XGDEN=collect(XGDEN_2, D);
>> XFNOM_2=expand(XFNOM1);
>> XFNOM=collect(XFNOM_2, D);
>> XFDEN_2=expand(XFDEN1);
```

```
>> XFDEN=collect(XFDEN_2, D);
>> XKNOM1=XFNOM*XNDEN+D^2*L*XNNOM;
>> XKDEN1=XFDEN*XNDEN;
>> XKNOM_2=expand(XKNOM1);
>> XKNOM=collect(XKNOM_2, D);
>> XKDEN_2=expand(XKDEN1);
>> XKDEN=collect(XKDEN_2, D);
>> XENOM1=D^3*L*XKDEN*XCNOM+XKNOM*XCDEN;
>> XEDEN1=XCDEN*XKDEN;
>> XENOM_2=expand(XENOM1);
>> XENOM=collect(XENOM_2, D);
>> XEDEN_2=expand(XEDEN1);
>> XEDEN=collect(XEDEN_2, D);
>> XMNOM1=(1-D^3*L*N)*XGNOM+...
(-D^6*L^2*N+D^3*L+D^2*L*N)*XGDEN;
>> XMDEN1=(1-D^3*L*N)*XGDEN;
>> XMNOM_2=expand(XMNOM1);
>> XMNOM=collect(XMNOM_2, D);
>> XMDEN_2=expand(XMDEN1);
>> XMDEN=collect(XMDEN_2, D);
>> XENOM_CH=char(XENOM);
>> fxenom=fopen('d:\xenom.txt','wt');
>> fwrite(fxenom,XENOM_CH,'char');
>> fclose(fxenom);
>> XEDEN_CH=char(XEDEN);
>> fxeden=fopen('d:\xeden.txt','wt');
>> fwrite(fxeden, XEDEN_CH,'char');
>> fclose(fxeden);
>> XENOM_START=(L^144*N^97)*D^239+(-13*L^143*N^96 -...
```

```
4*L^142*N^96)*D^238 + (L^143*N^100 - L^144*N^99 -...
L^143*N^98 - 38*L^143*N^97 + 74*L^142*N^95 +...
.............................................
+ 314*L^132*N^91 + 4744*L^132*N^90 +...
1327*L^132*N^89 - 2308*L^132*N^88 -...
681*L^131*N^88)*D^230;
>> XENOM_END=(3*L^19*N^17 + 41*L^19*N^16 +...
54*L^19*N^15 +34*L^19*N^14 +...
5*L^19*N^13 + L^19*N^12 - 14*L^18*N^16 +...
.............................................
3*L^8*N^7 + L^8*N^6)*D^17 + (L^8*N^7 -...
4*L^9*N^8 + 6*L^8*N^5)*D^16 + (-L^8*N^7)*D^15 +...
(L^7*N^6)*D^14;
>> XENOM = XENOM_START + XENOM_END;
>> XEDEN_START = (L^143*N^99)*D^234 +...
(- 13*L^142*N^98 - 5*L^141*N^98)*D^233 +...
(73*L^141*N^97 - 42*L^142*N^99 + 65*L^140*N^97 +...
.............................................
10*L^23*N^22 - 60*L^23*N^21 - 7*L^22*N^22)*D^45 +...
(20*L^22*N^20 - 7*L^23*N^23)*D^44 +...
(-3*L^22*N^22)*D^43 + (L^21*N^21)*D^42;
>> XEDEN = XEDEN_START + XEDEN_END;
>> XINOM1=D*L*(D^2*XENOM*XMDEN+XMNOM*XEDEN);
>> XIDEN1= XEDEN*XMDEN;
>> XINOM_2=expand(XINOM1);
>> XINOM=collect(XINOM_2, D);
>> XIDEN_2=expand(XIDEN1);
>> XIDEN=collect(XIDEN_2, D);
>> XINOM_CH=char(XINOM);
```

```
>> fxinom=fopen('d:\xinom.txt','wt');
>> fwrite(fxinom,XINOM_CH,'char');
>> fclose(fxinom);
>> XIDEN_CH=char(XIDEN);
>> fxiden=fopen('d:\xiden.txt','wt');
>> fwrite(fxiden, XIDEN_CH,'char');
>> fclose(fxiden);
>> XINOM_START= ((L^206*N^140 - L^208*N^143)*D^343 +...
(19*L^207*N^142 - L^207*N^141 + 6*L^206*N^142 -...
.....................................................
- 540*L^191*N^131 + 46602*L^191*N^130 +
3075*L^190*N^130)*D^333);
>> XINOM_END = (293*L^29*N^25 - 574*L^29*N^26 - ...
140*L^29*N^27 - 153*L^29*N^24 + 240*L^29*N^23 - ...
.....................................................
142071*L^189*N^132 - 7*L^188*N^136 - ...
568*L^188*N^134 - 31*L^188*N^133 + ...
19715*L^188*N^132 + 850*L^187*N^132)*D^325;
>> XIDEN_END = (924*L^50*N^50 - 33009*L^50*N^49 - ...
317055*L^50*N^48 - 685653*L^50*N^47 - ...
11*L^31*N^31)*D^63 + (28*L^31*N^29 - ...
.....................................................
4*L^32*N^32)*D^62 + (-5*L^31*N^31)*D^61 + ...
(L^30*N^30)*D^60;
>> TNOM1=D^4*L^2*N*XINOM*XBDEN;
>> TDEN1=XBNOM*XIDEN-D^2*L*N*XINOM*XBDEN;
>> TNOM_2=expand(TNOM1);
>> TNOM=collect(TNOM_2, D);
>> TDEN_2=expand(TDEN1);
```

```
>> TDEN=collect(TDEN_2, D);
>> TNOM_CH=char(TNOM);
>> ftnom=fopen('d:\tnom.txt','wt');
>> fwrite(ftnom,TNOM_CH,'char');
>> fclose(ftnom);
>> TDEN_CH=char(TDEN);
>> ftden=fopen('d:\tden.txt','wt');
>> fwrite(ftden,TDEN_CH,'char');
>> fclose(ftden);
```

Вміст файлу xenom.txt з розрахованим поліномом чисельника для стану скінченного автомату X_{enom} (R, L, N). Наведені початкові та останні доданки поліному

```
(L^144*N^97)*D^239 + (- 13*L^143*N^96 -
4*L^142*N^96)*D^238 + (L^143*N^100 - L^144*N^99 -
L^143*N^98 - 38*L^143*N^97 + 74*L^142*N^95 + 52*L^141*N^95
+ 6*L^140*N^95)*D^237 + (13*L^143*N^98 - L^143*N^97 -
6*L^143*N^96 - L^143*N^95 - 13*L^142*N^99 + 8*L^142*N^98 +
14*L^142*N^97 + 498*L^142*N^96 + 33*L^142*N^95 +
2*L^142*N^94 - 4*L^141*N^99 + 4*L^141*N^97 + 136*L^141*N^96
- 242*L^141*N^94 - 296*L^140*N^94 - 78*L^139*N^94 -
4*L^138*N^94)*D^236 +
.....................................................
+ (L^12*N^8 - 3*L^12*N^10 - 3*L^12*N^9 - 6*L^12*N^11 +
4*L^11*N^10 - 10*L^11*N^9 + 25*L^11*N^8 - L^11*N^7 +
6*L^11*N^6 + 10*L^10*N^9 - 3*L^10*N^8 - 4*L^10*N^7 -
4*L^10*N^6 - 20*L^10*N^5 - 2*L^9*N^8 - 18*L^9*N^6 +
6*L^9*N^5)*D^19 + (6*L^11*N^10 + 6*L^11*N^9 - 4*L^10*N^9 +
9*L^10*N^8 - 16*L^10*N^7 + 3*L^9*N^8 - L^9*N^7 + 6*L^9*N^6
```

```
+ 12*L^9*N^4)*D^18 + (4*L^10*N^9 + L^10*N^7 - L^9*N^8 -
L^9*N^7 - 8*L^9*N^6 - 3*L^8*N^7 + L^8*N^6)*D^17 + (L^8*N^7
- 4*L^9*N^8 + 6*L^8*N^5)*D^16 + (-L^8*N^7)*D^15 +
(L^7*N^6)*D^14
```

Вміст файлу xeden.txt з розрахованим поліномом знаменника передавальної функції скінченного автомату X_{eden} (R, L, N). Наведені початкові та останні доданки поліному

```
(L^143*N^99)*D^234 + (- 13*L^142*N^98 -
5*L^141*N^98)*D^233 + (73*L^141*N^97 - 42*L^142*N^99 +
65*L^140*N^97 + 10*L^139*N^97)*D^232 + (5*L^141*N^100 -
7*L^142*N^98 + 551*L^141*N^98 + 37*L^141*N^97 +
190*L^140*N^98 - 231*L^140*N^96 - 365*L^139*N^96 -
130*L^138*N^96 - 10*L^137*N^96)*D^231 +
.....................................................
+ (15*L^27*N^27 – 108*L^27*N^26 + 80*L^27*N^25 –
15*L^27*N^24+10*L^27*N^23+169*L^26*N^25+180*L^26*N^24+
385*L^26*N^23 - 215*L^26*N^22 - 203*L^25*N^25 –
7*L^25*N^23 - 200*L^25*N^22 - 905*L^25*N^21 –
60*L^24*N^23 + 425*L^24*N^22 + 960*L^24*N^18 +
21*L^23*N^23)*D^48 + (5*L^26*N^23 - 49*L^26*N^25 –
40*L^26*N^24 - 125*L^26*N^26 - 132*L^25*N^24 +
520*L^25*N^23 - 5*L^25*N^22 + 100*L^25*N^21 +
49*L^24*N^24 + 2*L^24*N^22 + 180*L^24*N^21 –
540*L^24*N^20 - 140*L^23*N^21)*D^47 + (15*L^25*N^25 +
17*L^25*N^24 - 10*L^25*N^23 - 10*L^24*N^23 –
120*L^24*N^22 + 21*L^23*N^23 + 180*L^23*N^19)*D^46 +
(29*L^24*N^24 + 5*L^24*N^22 + 10*L^23*N^22 –
60*L^23*N^21 - 7*L^22*N^22)*D^45 + (20*L^22*N^20 –
```

```
7*L^23*N^23)*D^44 + (-3*L^22*N^22)*D^43 + (L^21*N^21)*D^42
```

Вміст файлу xinom.txt з розрахованим поліномом чисельника для стану скінченного автомату X_{inom} (R, L, N). Наведені початкові та останні доданки поліному

```
(L^206*N^140 - L^208*N^143)*D^343 + (19*L^207*N^142 -
L^207*N^141 + 6*L^206*N^142 - 19*L^205*N^139 -
6*L^204*N^139)*D^342 + (58*L^207*N^143 - L^206*N^142 -
165*L^206*N^141 + 18*L^206*N^140 + L^205*N^143 -
115*L^205*N^141 - 49*L^205*N^140 - 15*L^204*N^141 +
165*L^204*N^138 + 114*L^203*N^138 + 15*L^202*N^138)*D^341
.....................................................
+ 280*L^20*N^12 + 21*L^19*N^17 - 7*L^19*N^16)*D^41
+ (3*L^22*N^17 - 35*L^22*N^19 - 18*L^22*N^18 -
76*L^22*N^20 + 23*L^21*N^19 - 70*L^21*N^18 +
291*L^21*N^17 - 3*L^21*N^16 + 42*L^21*N^15 +
33*L^20*N^18 - 39*L^20*N^16 + 36*L^20*N^15 -
276*L^20*N^14 - 6*L^19*N^17 - 98*L^19*N^15 +
14*L^19*N^14)*D^40 + (4*L^21*N^19 + 13*L^21*N^18 -
6*L^21*N^17 - 5*L^20*N^18 + 7*L^20*N^17 - 50*L^20*N^16 +
21*L^19*N^17 - 3*L^19*N^16 + 14*L^19*N^15 +
84*L^19*N^13)*D^39 + (23*L^20*N^18 + 3*L^20*N^16 -
3*L^19*N^17 + 3*L^19*N^16 - 44*L^19*N^15 - 7*L^18*N^16 +
L^18*N^15)*D^38 + (L^18*N^16 - 5*L^19*N^17 +
14*L^18*N^14)*D^37 + (-3*L^18*N^16)*D^36 +
(L^17*N^15)*D^35
```

Вміст файлу xiden.txt з розрахованим поліномом знаменника передавальної функції скінченного автомату X_{iden} (R, L, N). Наведені початкові та останні доданки поліному

```
(L^204*N^142)*D^335 + (- 19*L^203*N^141 -
7*L^202*N^141)*D^334 + (164*L^202*N^140 - 59*L^203*N^142 +
133*L^201*N^140 + 21*L^200*N^140)*D^333 +
.....................................................
+ (476*L^36*N^36 - 87*L^36*N^35 +
322*L^36*N^34 - 35*L^36*N^33 + 21*L^36*N^32 +
761*L^35*N^34 - 1316*L^35*N^33 + 777*L^35*N^32 -
805*L^35*N^31 - 660*L^34*N^34 - 16*L^34*N^32 -
1120*L^34*N^31 - 735*L^34*N^30 - 140*L^33*N^32 +
1547*L^33*N^31 + 2912*L^33*N^27 + 55*L^32*N^32)*D^66 +
(7*L^35*N^32 - 117*L^35*N^34 - 49*L^35*N^33 -
316*L^35*N^35 - 199*L^34*N^33 + 1540*L^34*N^32 -
7*L^34*N^31 + 196*L^34*N^30 + 44*L^33*N^33 + 3*L^33*N^31 +
364*L^33*N^30 - 1820*L^33*N^29 - 308*L^32*N^30)*D^65 +
(24*L^34*N^33 - 44*L^34*N^34 - 28*L^34*N^32 -
42*L^33*N^32 - 84*L^33*N^31 + 55*L^32*N^32 +
364*L^32*N^28)*D^64 + (60*L^33*N^33 + 7*L^33*N^31 +
14*L^32*N^31 - 140*L^32*N^30 - 11*L^31*N^31)*D^63 +
(28*L^31*N^29 - 4*L^32*N^32)*D^62 + (-5*L^31*N^31)*D^61 +
(L^30*N^30)*D^60
```

Вміст файлу tnom.txt з розрахованим поліномом чисельника передавальної функції скінченного автомату T_{nom}(R, L, N). Наведені початкові та останні доданки поліному

```
(L^269*N^183 - L^271*N^186)*D^448 + (25*L^270*N^185 -
```

```
L^270*N^184 + 8*L^269*N^185 - 25*L^268*N^182 -
8*L^267*N^182)*D^447 + (75*L^270*N^186 - L^269*N^185 -
292*L^269*N^184 + 24*L^269*N^183 + L^268*N^186 -
201*L^268*N^184 - 64*L^268*N^183 - 28*L^267*N^184 +
292*L^267*N^181 + 200*L^266*N^181 +
28*L^265*N^181)*D^446 + (14*L^270*N^185 + L^270*N^184 -
9*L^269*N^187 - 1883*L^269*N^185 + 9*L^269*N^184 -
2*L^269*N^183 - 568*L^268*N^185 + 25*L^268*N^184 +
2115*L^268*N^183 - 282*L^268*N^182 - L^268*N^181 -
25*L^267*N^185 + 16*L^267*N^184 + 2362*L^267*N^183 +
1616*L^267*N^182 + 63*L^267*N^181 + 2*L^267*N^180 -
8*L^266*N^185 + 708*L^266*N^183 + 516*L^266*N^182 -
2116*L^266*N^180 + 56*L^265*N^183 - 2336*L^265*N^180 -
700*L^264*N^180 - 56*L^263*N^180)*D^445 +
.....................................................
+ (5*L^33*N^26 - 95*L^33*N^28 - 25*L^33*N^27 -
211*L^33*N^29 + 50*L^32*N^28 - 142*L^32*N^27 +
997*L^32*N^26 - 5*L^32*N^25 + 110*L^32*N^24 +
20*L^31*N^27 + 7*L^31*N^26 - 106*L^31*N^25 +
140*L^31*N^24 - 1140*L^31*N^23 - 10*L^30*N^26 -
242*L^30*N^24 + 22*L^30*N^23)*D^62 + (20*L^32*N^27 -
49*L^32*N^28 - 20*L^32*N^26 - 2*L^31*N^27 -
11*L^31*N^26 - 12*L^31*N^25 + 55*L^30*N^26 -
5*L^30*N^25 + 22*L^30*N^24 + 220*L^30*N^22)*D^61 +
(50*L^31*N^27 + 5*L^31*N^25 - 5*L^30*N^26 +
7*L^30*N^25 - 112*L^30*N^24 - 11*L^29*N^25 +
L^29*N^24)*D^60 + (L^29*N^25 - 2*L^30*N^26 +
22*L^29*N^23)*D^59 + (-5*L^29*N^25)*D^58 +
(L^28*N^24)*D^57
```

Вміст файлу tden.txt з розрахованим поліномом знаменника передавальної функції скінченного автомату T_{den} (R, L, N). Наведені початкові та останні доданки поліному

```
(L^270*N^186 - L^268*N^183)*D^446 + (L^269*N^187 -
L^270*N^185 - 25*L^269*N^185 + L^269*N^184 -
8*L^268*N^185 + 25*L^267*N^182 +
8*L^266*N^182)*D^445 + (25*L^269*N^184 -
75*L^269*N^186 - 25*L^268*N^186 + L^268*N^185 +
300*L^268*N^184 - 24*L^268*N^183 - 9*L^267*N^186 +
201*L^267*N^184 + 64*L^267*N^183 + 28*L^266*N^184 -
292*L^266*N^181 - 200*L^265*N^181 -
28*L^264*N^181)*D^444 + (L^269*N^186 - L^269*N^188 -
L^269*N^189 + 58*L^269*N^185 - L^269*N^184 -
64*L^268*N^187 + 1883*L^268*N^185 - 9*L^268*N^184 -
290*L^268*N^183 + 860*L^267*N^185 - 25*L^267*N^184 -
2315*L^267*N^183 + 282*L^267*N^182 + L^267*N^181 +
225*L^266*N^185 - 16*L^266*N^184 - 2390*L^266*N^183 -
1616*L^266*N^182 - 63*L^266*N^181 - 2*L^266*N^180 +
36*L^265*N^185 - 708*L^265*N^183 - 516*L^265*N^182 +
2116*L^265*N^180 - 56*L^264*N^183 + 2336*L^264*N^180 +
700*L^263*N^180 + 56*L^262*N^180)*D^443 +
.....................................................
+ (211*L^32*N^29 + 95*L^32*N^28 +
25*L^32*N^27 - 5*L^32*N^26 - 50*L^31*N^28 +
142*L^31*N^27 - 997*L^31*N^26 + 5*L^31*N^25 -
110*L^31*N^24 - 20*L^30*N^27 - 7*L^30*N^26 +
106*L^30*N^25 - 140*L^30*N^24 + 1140*L^30*N^23 +
10*L^29*N^26 + 242*L^29*N^24 - 22*L^29*N^23)*D^60 +
```

```
(49*L^31*N^28 - 20*L^31*N^27 + 20*L^31*N^26 +
2*L^30*N^27 + 11*L^30*N^26 + 12*L^30*N^25 - 55*L^29*N^26 +
5*L^29*N^25 - 22*L^29*N^24 - 220*L^29*N^22)*D^59 +
(5*L^29*N^26 - 5*L^30*N^25 - 50*L^30*N^27 - 7*L^29*N^25 +
112*L^29*N^24 + 11*L^28*N^25 - L^28*N^24)*D^58 +
(2*L^29*N^26 - L^28*N^25 - 22*L^28*N^23)*D^57 +
(5*L^23*N^20)*D^51 + (-L^22*N^20)*D^50
```